MANAGING HETEROGENEOUS RESOURCES FOR DYNAMIC ENERGY-EFFICIENT SENSING

BY

DAVID MOSES JUN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Doctoral Committee:

Professor Douglas Jones, Chair
Assistant Professor Maxim Raginsky
Professor Naresh Shanbhag
Professor Venu Veeravalli

ABSTRACT

Next-generation embedded devices are expected to pervasively extract information from the world around us. In particular, growing interest in mobile devices beyond the smart phone imply that the need for context awareness and information extraction is more important than ever before.

This thesis considers the optimization in the utilization and design of sensing resources on a single device, in order to extract high-quality information at energy consumption rates that are dramatically lower than what is expected today. The key insight is to match device resources to the information available in the environment, which requires active resource management along with a diverse set of efficient hardware and software components.

The problem of matching device resources to the information available in dynamic environments is fundamentally difficult because the amount of effort and energy spent on sensing is typically related to the quality of the data acquired, which determines the accuracy of understanding and predicting the state of the environment. This thesis develops the analysis required to understand optimal trade-offs between performance and energy consumption for a system with a given set of sensing and processing resources. The utilization problem is mapped to a partially observable Markov decision process (POMDP) and the appropriate mapping is derived in order to leverage state-of-the-art POMDP numerical solvers to generate optimal resource-scheduling policies.

Developing a tool to determine the optimal achievable performance/energy trade-off for a given set of resources enables system designers to understand the inefficiencies of heuristic sensing strategies, and also to propose more aggressive trade-offs for new sensors, signal chains, and algorithms. We present a case study of our approach to system design, using an acoustic wildlife monitoring task as the application driver. A hidden Markov model (HMM) approach to pattern recognition is adopted, where event arrivals and bird songs are modeled using a 30-state Markov chain, with transition parameters learned from data.

In order to characterize the signal fidelity and energy costs associated with sensing and processing resources, the CheetahCub testbed is developed, which combines a TI MSP430 low-power microcontroller with dynamic voltage scaling driven by algorithmic processing demands. The testbed achieves 16× energy scalability, ranging from simple energy calculations to more sophisticated signal processing. We also discuss the EFM testbed, based on the 32-bit ARM Cortex-M3 processor, which proves to be 2.5× more efficient than the CheetahCub testbed when high processing capabilities are required.

In both testbeds, the data acquisition front-end is shown to be the energy bottleneck. Thus, a novel signal chain for acoustic sensing is proposed. The proposed signal chain replaces the typical preamplifier circuit and ADC with a digital noise floor tracker and analog comparator. The sensing package consumes 10× less energy than a traditional microphone circuit but at the expense of degraded signal fidelity.

Observation models are learned from data for the sensing actions developed in this thesis. The procedure for mapping the problem to a POMDP to generate optimal scheduling policies is demonstrated, and our approach to system design is validated by evaluating the optimal performance/energy trade-off

achieved by our system. Including the proposed nano-power acoustic sensor demonstrates an order of magnitude reduction in total system energy consumption, relative to an efficient approach based on cascading signal models for energy-aware detection. This process demonstrates the powerful synergy achieved by utilizing theory to enable systematic evaluation of aggressive, innovative sensor-design trade-offs. Our optimal scheduling framework is also used to study the efficiency of an intuitive wakeup mechanism, demonstrating that heuristic design may be missing out on 2×-4× energy savings, compared to optimal scheduling.

In closing, this thesis challenges conventional wisdom that devices must be designed to sleep in ultra low-power sensing applications. We demonstrate that this does not necessarily have to be true, if one can combine innovative sensing hardware with clever resource management to match the information available in dynamic environments. This will become increasingly relevant as emerging applications in wearable computing and the Internet of Things demand high-quality information and context awareness.

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Professor Doug Jones, for enthusiastically supporting my research interests and providing invaluable guidance and intuition. I am indebted to Dr. Todd Coleman who first suggested and then helped me to apply stochastic control techniques to my research problems. I would like to thank Dave Cohen and Long Le for all of the lengthy, yet fruitful discussions we had at the awkwardly sized whiteboard.

I would like to acknowledge my lab mates, Erik, Cagdas, Dave, Long, and Doom, for making otherwise boring days at the office much more entertaining. I owe my deepest gratitude to Dr. Miri Kim, who has been supportive and understanding throughout every step of this process. Finally, I would like to thank my parents, who have supported me from a distance with food and their prayers.

This work was supported in part by the Multiscale Systems Center (MuSyC) and the TerraSwarm Research Center of the Focus Center Research Program (FCRP), a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1
1.1 Motivation 1
1.2 Opportunities to Save Device Energy 2
1.3 Managing Resources 3
1.4 Energy-Efficient Heterogeneous Sensing Resources 4
1.5 Thesis Contributions 5

CHAPTER 2 RESOURCE MANAGEMENT FOR INFORMATION-RICH SENSING 8
2.1 Motivation for a Principled Framework 9
2.2 Problem Formulation 11
2.3 Review of POMDP, Methods, and Sensor Management 20
2.4 On Numerical Tractability 33
2.5 Establishing the Principle of Optimality 36
2.6 An Exact Direct Solver 40
2.7 Efficient Approximate Solution using Point-based Solvers . . . 42
2.8 An Improved Lower Bound 44

CHAPTER 3 ENERGY-EFFICIENT SENSING 50
3.1 Application Driver: Acoustic Wildlife Monitoring 50
3.2 Acoustic Model 50
3.3 CheetahCub Testbed 55
3.4 EFM Testbed 59
3.5 A Nano-power Acoustic Sensor 61

CHAPTER 4 SYSTEM DESIGN 66
4.1 Temporal Dynamics 66
4.2 Application Goals 70
4.3 Sensing Actions and Energy Consumption 73
4.4 Observation Models 74
4.5 Run-time Operation 76

CHAPTER 5 PERFORMANCE EVALUATION 78
5.1 Evaluation Procedure 78
5.2 Setting the Discount Factor 79
5.3 Synergistic Design 79
5.4 Comparing Different Testbeds 81
5.5 The Efficiency of the Wakeup Mechanism 81

CHAPTER 6 CONCLUSIONS AND FUTURE DIRECTIONS 84
6.1 Conclusions 84
6.2 Future Directions 87

APPENDIX A CASCADING SIGNAL MODELS FOR ENERGY-AWARE DETECTION 90
A.1 Problem Formulation 91
A.2 Solution Characterization 95
A.3 Conclusions 101
A.4 Proof of Theorem 2 102

REFERENCES 106

CHAPTER 1

INTRODUCTION

1.1 Motivation

Next-generation embedded devices are expected to pervasively extract information from the world around us. In particular, growing interest in mobile devices beyond the smart phone (e.g., wearables, IoT) imply that the need for context awareness and information extraction is more important than ever before.

Advances in processing technology have created devices that are capable of handling more computations being pushed out to these low-power processors [1, 2], but current battery technology limits the utilization of such capabilities. To illustrate the point, microcontrollers and small battery-powered embedded devices have long been designed with the principle to "do nothing well". As a result, even though they have a 16 or 32-bit architecture capable of millions of instructions per second, they are optimized to sleep and have been used only for processing simple tasks.

Our goal is to deliver sophisticated sensing while consuming energy that is dramatically lower than what is expected today. Our thesis is that this can be achieved not only by "doing nothing well", but more generally, by "doing more, doing less, and doing it better".

This raises interesting and hard system-level challenges, because although doing nothing well can scale with processor technology, dynamically doing

more or less raises questions such as "when?" and "by how much?" We propose that the answers to these questions require active resource management and system-level objectives, along with hardware and software that scales gracefully with the workload (i.e., energy efficient when it is doing nothing, doing a little, and doing a lot).

1.2 Opportunities to Save Device Energy

To expound upon the mantra of "doing more, doing less, and doing it better", we propose to accomplish our goals by designing systems that can efficiently match processing demands to the activity level and information available in the environment. Opportunities to reduce device energy are created by the *heterogeneity* in the physical environment that is being sensed.

Monitoring for the arrival of physical events represents a broad class of cyber-physical sensing applications that exhibit this heterogeneity. Several applications include always-listening voice command recognition, daily-activity monitoring, and wildlife tracking. In these applications, the physical events have inertia and temporal structure that can be exploited and predicted to save device energy. In particular, enormous energy savings can be had in applications where the events of interest are rare. In this thesis, we formalize this intuition, building realistic stochastic models of the physical environment, thus explicitly modeling the opportunities to save device energy.

1.3 Managing Resources

The problem of matching device resources to the information available in dynamic environments is fundamentally difficult because the amount of effort and energy spent on sensing is typically related to the quality of the data acquired. The quality of the data, in addition to the accuracy and complexity of the signal processing algorithms applied to that data, influence the ability to understand the true state of the environment. The problem can become compounded because it is based on this inferred understanding that future allocations of sensing resources must be made. Thus, not accounting for the inherent uncertainty in sensors or limitations in processing algorithms can lead to inefficient resource scheduling.

Given a set of sensing and processing resources, there are many commonly used heuristic techniques for energy-efficient sensing. In applications for continuous monitoring of rare events, an intuitive strategy is to implement a wake-up mechanism. Here, a low-power, low-complexity sensor runs continuously, looking for potentially interesting activity. When activity is detected, more sensing and processing resources are awakened to analyze the data more carefully. Abstractly, wakeup mechanisms, cascade architectures, and decision trees, to name a few, share this basic idea, which is to increase sensing and processing complexity in response to what is being discovered. The main issue with heuristics is the lack of a systematic approach to design and analysis. Issues include *implicit* assumptions about application characteristics, heuristic scheduling architectures based on hard decisions, no systematic handling of uncertainty, and no notion of the optimality of a particular architecture, making it difficult to systematically identify weaknesses in design.

To handle all of these shortcomings, this thesis lays down the foundation

for constructing resource scheduling schemes that make optimal decisions under uncertainty, where the uncertainty arises from the explicitly modeled application dynamics and signal fidelity of different sensors and processing algorithms. The value achieved by optimal resource schedulers provides a bound on the performance/energy trade-off, which is shown to be a useful guide for system design.

1.4 Energy-Efficient Heterogeneous Sensing Resources

When considering the problem of managing device resources, no distinction is made between sensing and processing resources; they are considered abstractly as any *sensing action* that is capable of generating observations. Practically, a sensing action consists of sensors, signal chains, and processing algorithms.

Physical sensors and their analog signal chains (e.g., preamplifiers and ADCs) are responsible for converting physical phenomena into a digital representation (e.g., sound pressure waves into a wave file). By being flexible and designing sensors that trade off signal fidelity for power consumption, a suite of sensors can potentially be utilized. Similarly, even for a single sensor, a variety of processing algorithms can be utilized to extract information from noisy data. Algorithms range from very simple (e.g., computing signal energy) to sophisticated statistical signal processing methods.

To characterize the effective signal-to-noise ratio (SNR) and energy consumption of different combinations of sensors, signal chains, and processing algorithms, significant emphasis is placed on actually building energy-efficient embedded systems. This is necessary not only because power consumption profiles are not available in the literature or datasheets, but also because pro-

filing realistic power numbers for the various hardware and software components enables us to pinpoint bottlenecks and inefficiencies in existing system designs, pointing to the areas that require further research efforts.

1.5 Thesis Contributions

This thesis presents a new approach to the system-level design of energy-efficient sensing devices that are to be used in dynamic, noisy environments. Our strategy separates the optimization of the utilization of sensing resources from the design of these resources. The implication is that system designers can understand the application-level performance / energy consumption trade-off for any particular sensing resource, without having to worry about how and when to use it. This in turn enables rapid design-space exploration for resource design.

The separation of utilization from design is achieved by formulating the resource-utilization problem as an appropriate functional optimization. We rigorously formulate the problem of creating optimal resource-utilization strategies for time-varying monitoring applications as a partially observable Markov decision process (POMDP). Using state-of-the-art results from stochastic control theory and machine learning, we achieve the following:

- explicitly model random event arrivals and account for temporal correlations of events of interest,
- explicitly account for the uncertainty introduced by noisy sensors and processing algorithms,
- numerically compute non-myopic optimal strategies that schedule sensing and processing resources to balance inference performance with en-

ergy consumption.

To illustrate our approach to system-level design, we present a case study using an acoustic wildlife monitoring task as the application driver. Our approach to pattern recognition uses hidden Markov models (HMM), with techniques borrowed from the speech recognition literature, and is a natural fit for the abstract POMDP approach to resource management.

To explore resource design, we built an MSP430 testbed with dynamic voltage and frequency scaling capabilities to support hugely energy-scalable processing. This testbed aims to optimize the energy that is spent, and quantifies the low power consumption that is possible to achieve, even on today's commercially available components. Furthermore, we utilized an ARM Cortex-M3 testbed, which was used to study the efficiency of a 32-bit ARM microcontroller designed to deliver performance in low-power applications.

Due to the data acquisition front-end being the energy bottleneck in both testbeds, we propose a novel signal chain for acoustic sensing, which has immediate impact in always-on applications. This ultra low-power sensor is too noisy to be useful on its own or as the always-on sensor in a wakeup mechanism, but becomes an invaluable resource with proper scheduling.

Our approach to system design is quantified and compared against the performance of efficient scheduling architectures, including a wakeup mechanism and an energy-aware cascade of signal models. The optimal policy generated by the POMDP solver demonstrates the achievable performance/energy trade-offs for a given system design; this automated optimal use of device resources accelerates design-space exploration, pointing out the system components that need to be redesigned or further optimized.

The approach to system design developed in this thesis validates the viability of delivering high-performance, information-rich acoustic sensing in ultra

low-power applications, a space in which severe constraints on battery capacity have resulted in applications with very limited sensing capabilities. The principled approach we develop enables innovative design trade-offs, promising to achieve high energy efficiency in dynamic sensing applications.

CHAPTER 2

RESOURCE MANAGEMENT FOR INFORMATION-RICH SENSING

The theoretical and numerical techniques developed in this chapter result in new analysis tools for system designers to manage how device resources are utilized for energy-efficient sensing in dynamic information-rich environments. Our approach builds upon theoretical techniques from stochastic control and numerical algorithms from machine learning.

Beginning with motivation for a principled approach in Section 2.1, we are the first to formulate the problem of a *controlled* generative approach to energy-efficient pattern recognition in Section 2.2, where sequential control actions map to both sensing decisions and inference decisions. The problem formulation is organized such that the review of partially observable Markov decision processes (POMDP) in Section 2.3, which is required to understand the long-term implications of sequential decisions, follows easily. Given the fundamental difficulty of POMDPs and limitations of current technology, Section 2.4 refines our problem formulation such that general tools to solve our class of problems can be developed.

The final four sections of the chapter develop the solution to the problem, signifying our contributions to the field. Section 2.5 shows how the non-classical flow of information of our problem follows dynamic programming principles. Section 2.6 presents an exact numerical solver that exploits the fact that inference decisions do not impact the future evolution of the belief state. As exact solvers are known to not scale with problem size, Section 2.7

maps our class of problems into a form that can use existing point-based general-purpose numerical solvers. Finally, Section 2.8 derives an improved lower bound to the optimal value function over a bound that is commonly utilized in the field; our derivation follows from exploiting problem characteristics that general-purpose POMDPs do not share.

2.1 Motivation for a Principled Framework

The wakeup mechanism is a popular intuitive mechanism for the design of an energy-efficient sensing architecture. Device energy consumption is reduced by running an efficient front-end [3, 4] that triggers subsequent stages of processing only when something interesting is detected. [5] presents an energy-efficient strategy for multi-modal detection of natural events that can be characterized as rare, random, and ephemeral. Their strategy consists of hierarchical sensing and processing, where stages of more complex processing are triggered based on information collected from sensors. [6] uses a hierarchical strategy in what it calls the tiered wake-up network. A decision tree is designed to fuse multi-modal sensors.

A weakness in all of these systems is that they are heuristically designed. Heuristic architectures are not inherently bad designs, but issues include:

1. *Manual exploitation of problem characteristics.*

   The issue is that application designers make *implicit* assumptions about the temporal structure of the underlying event of interest and exploit these characteristics to *manually* design sensing architectures to achieve energy efficiency. For example, a wakeup mechanism inherently assumes a certain degree of temporal correlation, as a detection in the current frame implies that the event of interest will still be detectable

in subsequent frames. While this is manageable for simple temporal structure, this approach does not generalize for complex dynamics.

2. *Manual design of sensing and processing architecture, which leads to artificial constraints and auxiliary design specifications that may not directly contribute to application goals.*

   As an example, consider the ultra low-power hardware voice activity detector (VAD) proposed to be used as a wakeup mechanism in [3]. The efficient VAD is used to detect speech activity and wake up a sensor network for further speech processing tasks (e.g., speech recognition). The VAD consists of a microphone and a simplified zero-crossing algorithm implemented in hardware. Because the wakeup mechanism is fixed, researchers are forced to optimize for standard detection metrics (minimizing missed detections and false alarms), since further action is triggered by the VAD's decisions. Unfortunately, there is no notion of what the application's final goals are, which could potentially lead to a very different design trade-off. Furthermore, a wakeup mechanism does not make full use of feedback. The architecture does not allow for information that is extracted downstream in the inference task to propagate back to make the front-end hardware more intelligent.

3. *In general, there is no way to answer the question of optimality.*

   That is, does a particular heuristically designed system leave energy savings on the table? If so, how much? Without a principled framework, this is a vague question and hard to answer because it depends on the inference task, dynamics in the level of physical activity, and quality of sensors and processing algorithms.

In this chapter, we propose a systematic framework that constructs opti-

mally energy-efficient scheduling architectures that determine when and how to use the sensing and processing resources available on a device. Application goals and energy consumption define the notion of optimality, and the formulation explicitly accounts for the dynamics of the physical event, activity level, and uncertainty arising from noisy environments, sensors and processing algorithms. The optimal schedulers respond to the current dynamics, leveraging everything that is observed and information that is gained to make the most informed decision about what resources to utilize.

2.2 Problem Formulation

The problem of designing energy-efficient sensing/processing architectures for inference applications is formulated as a sequential decision process under uncertainty. In this section, we make the transition from a motivation for a principled framework to a rigorous formulation that enables the construction of optimal sensing architectures.

2.2.1 A generative approach to pattern recognition

We begin by describing the nature of the application-level inference tasks considered. In particular, we focus on pattern recognition problems. The processing steps in these problems can be described generically by the block diagram in Fig. 2.1.

For example, in voice command recognition, the goal is to match patterns of acoustic data to the particular words that a system is trained to recognize. In activity monitoring, the goal may be to infer the number of steps taken by looking for repeating patterns in accelerometer data. In acoustic wildlife monitoring, the goal may be to find a particular time-frequency sequence of

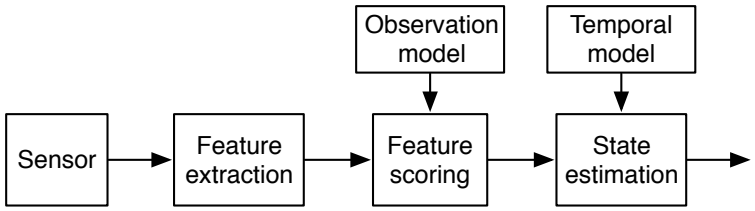


Figure 2.1: A block diagram illustrating the pattern recognition problem. This block diagram was adapted from [7, Ch 2] used to describe large vocabulary continuous speech recognition.

observations that matches a known bird song.

We adopt a generative approach to the pattern recognition problem, where we assume there exist stochastic models for both the observation and temporal models of Fig. 2.1. The temporal model explains how physical phenomena evolve over time, and the observation model explains how observations are generated from noisy data. In statistical signal processing, these model parameters are typically learned from training data, and at run-time, Bayes' rule is used to form the posterior distribution of the states given a sequence of observations. In this thesis, we assume the generative process is a hidden Markov model (HMM), as shown in Fig. 2.2. In particular, we assume

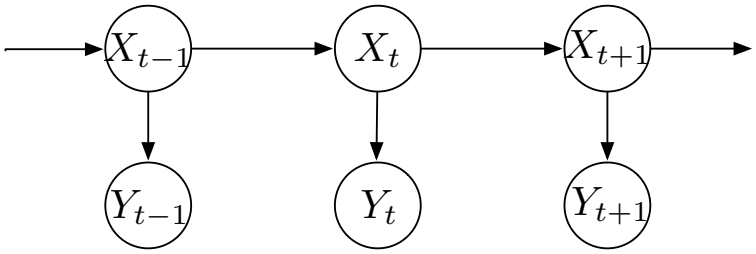


Figure 2.2: Temporal sequence of an HMM. States evolve according to a Markov chain, and observations are conditionally independent given the state.

that states evolve according to a discrete-time first-order Markov chain, and observations are conditionally independent, given the current state. More specifically, denote $\bar{\mathcal{X}}$ as a finite state space, with a transition kernel that

satisfies the Markov property:

$$Pr(X_{t+1} = x' | X_t = x_t) = Pr(X_{t+1} = x' | X^t = \{x_0, \ldots, x_t\}) \tag{2.1}$$

An HMM observation, as illustrated in Fig. 2.3, is the output of the feature extraction step, which generally operates on a block of raw sensor data. Thus, the rate of observations is typically slower than that of the physical

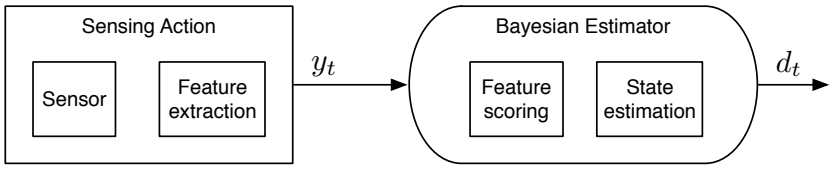


Figure 2.3: An HMM observation is the output of the feature extraction block in a typical pattern recognition problem.

sensor's sampling rate. For example, consider a speech application where a microphone is sampled at 16.384 kHz, and MFCC features are extracted on a block size of 512 samples, with 50% overlap. Then, HMM observations (i.e., MFCC features) are generated at a rate of 64Hz.

2.2.2 Event arrivals

In Fig. 2.1, the arrival of a potential event of interest is typically assumed to have been taken care of in a prior step, for example, by using a VAD mechanism. Thus, although the temporal model may include a silent state before and after the pattern to account for the fact that the segmentation process is not perfect, there are no extra states to model the arrival rate of the event.

Because we are interested in managing *all* sensing resources (including any low-power wake-up mechanisms) in order to achieve application goals,

we propose to incorporate the arrival of events in the temporal models by hierarchically augmenting the state space as shown in Fig. 2.4.

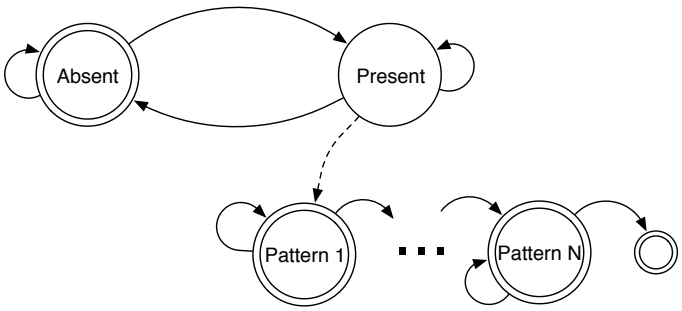


Figure 2.4: Hierarchically augmenting the temporal model utilized in pattern recognition to model event *arrivals*.

For example, if the waiting time for an event arrival is geometrically distributed, then a single additional state needs to be augmented. More general duration distributions require more states, and this issue is discussed in Chapter 4.

A hierarchical Markov chain can be enumerated into a flattened Markov chain [8]; denote the flattened state space as $\mathcal{X}$. In the example where event arrivals are geometrically distributed, $|\mathcal{X}| = |\bar{\mathcal{X}}| + 1$.

2.2.3 State estimation task

State estimation is typically the result of solving an optimization criterion, whether it is minimizing mean squared error or maximizing the a-posteriori probability of a sequence of observations. Because acquiring observations is only an intermediate step in inference problems, this is motivation to combine the two problems of 1) the intermediate goal of making scheduling decisions about sensing actions and 2) the ultimate goal of making inference decisions about the state of nature.

To make things explicit, we define a space of inference decisions $\mathcal{D}$ such that if decision $d \in \mathcal{D}$ is made when the state is $x \in \mathcal{X}$, a reward of $\bar{r}(x, d)$ is received. For example, $\mathcal{D} = \mathcal{X}$ and $\bar{r}(d, x) = 1$ iff $d = x$ corresponds to a sequential MAP state estimator. Generalizing the task of state estimation to decisions about inference is useful when the decision corresponds to high-level abstract summary information about the state of the world; we illustrate the utility of this generalization in Chapter 4 with examples that achieve different application goals simply by specifying different decision spaces and corresponding reward functions.

2.2.4 Sensing actions

The definition of a sensing action is naturally defined to be the entire sensing and processing chain used to generate an HMM observation, from the hardware sensor to the extracted features. The notion of a sensing action enables us to define *multiple* sensing actions. Continuing a previous example, an alternative sensing action could be a microphone sampled at 8.192kHz followed by calculating the total energy accumulated over 15.6ms. Here, the physical sensor is run at half the sampling rate and a simpler, lower dimensional, less processor-intensive feature is extracted.

Let $\mathcal{S}$ be the finite space of sensing actions. The observation function: $\bar{\mathcal{O}} : \mathcal{X} \times \mathcal{S} \times \mathcal{Y} \rightarrow [0, 1]$ characterizes the signal fidelity of each sensing action $s \in \mathcal{S}$:

$$\bar{o}(x, s, y) = Pr(y|x, s) = Pr(y|x, x', s) \tag{2.2}$$

where it is assumed that for all sensing actions, observations are conditionally independent given the state.

The motivation for introducing multiple sensing actions is clear; data collection is only an intermediate step for Bayesian inference, where the system-level goal is related to accurate state estimation. The key insight that many heuristic systems implicitly assume is that lower fidelity data collected using less energy may have the same value of information if nothing interesting is physically occurring or if the data SNR is very high.

2.2.5 Energy consumption

The energy consumption assigned to a sensing action includes all of the sensing, processing, and memory resources utilized in Fig. 2.1. This includes the hardware (e.g., preamplifiers, ADC, accelerators) as well as the software (e.g., preprocessing algorithms, DSP libraries, math functions).

The energy associated with each sensing action includes not only resources consumed for data collection, but also for inference, which include feature scoring and state estimation in Fig. 2.1. The computational complexity and memory requirements associated with feature scoring directly depend on the complexity of the observation model associated with the sensing action. The energy consumed by feature scoring can vary drastically, from being a trivial overhead for small discrete observation spaces, to being the most computationally intensive component of the processing chain (e.g., multi-dimensional features modeled using GMMs [7]).

State estimation consists mainly of computing Bayes' rule and, given feature scores, depends mainly on the complexity of the state space and temporal models. In particular, discrete states are simple to update, while continuous states require Kalman filtering under Gaussian models, or particle filtering under more general temporal models.

It is important to account for all of the energy consumed as a result of selecting a particular sensing action. As an extreme example, consider an application where running a particle filter completely dominates the energy consumption, relative to sensing, feature extraction, or scoring. In this case, dynamically scheduling sensing actions will have very little impact on controlling the total amount of energy consumed.

Denote the energy-consumption function $\mathcal{C} : \mathcal{X} \times \mathcal{S} \to \mathcal{R}_+$, where $c(x, s)$ is the energy consumed by sensing action s, which can depend on the state of the world, x.

In this thesis, we do not address the important issue of dynamically profiling or measuring the energy consumption of the hardware and software components during runtime [9]. Our solution is to implement sensing actions on embedded systems and profile the average power consumption offline. This process is demonstrated in Chapter 3. Our approach to system design leads to useful insight in terms of identifying system bottlenecks and pushing the limits on the achievable energy savings.

Finally, we do not explicitly account for the energy consumed by the resource manager, which is what we are trying to construct using optimization methods. For deployment, we believe lightweight heuristic scheduling policies will need to be developed. The solutions we construct in this thesis represent the optimal performance/energy trade-offs that are achievable for a given system, and should be used to guide heuristic design.

2.2.6 Decision making: optimal sensing architectures

As illustrated in Fig. 2.5, introducing multiple sensing actions requires managing which sensing action to execute at any given time. Due to energy

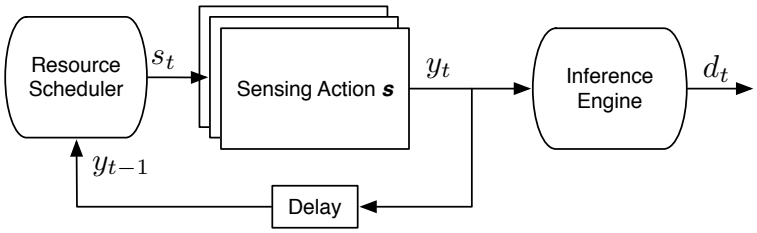


Figure 2.5: A block diagram of the decision process. At the beginning of time t, the Resource Scheduler uses all of the information collected thus far, including observation y_{t-1}, to make an informed decision about what sensing action, s_t, to take now. After an observation is generated, the Inference Engine uses all of the observations and actions taken up to time t to make an informed inference decision, d_t.

being a limited resource, we require that any management scheme be sequential – one cannot go back in time and "unuse" a sensor and "unsee" the observation. Thus, the resource scheduler should use all of the accumulated information thus far to make the most informed decision when choosing the next sensing action. This sensing decision should be made by predicting the long-term value of each action, accounting for the potential information that can be extracted along with the energy that is to be consumed. After an observation is collected, all of the accumulated information should be used to make an inference decision, d_t.

To incorporate the HMM assumptions made earlier in the section, Fig. 2.6 shows a more specific block diagram showing the conditional dependencies. In particular, at the beginning of time t, the state evolves from x_{t-1} to x_t; this evolution is hidden from the system. The information available to the system consists of the following information:

$$I_t^s = \{\mathbf{b}_0, d_0, s_1, y_1, d_1, \ldots, s_{t-1}, y_{t-1}, d_{t-1}\} \tag{2.3}$$

where $\mathbf{b}_0$ represents the initial belief about the state x_0. Then, the sensing

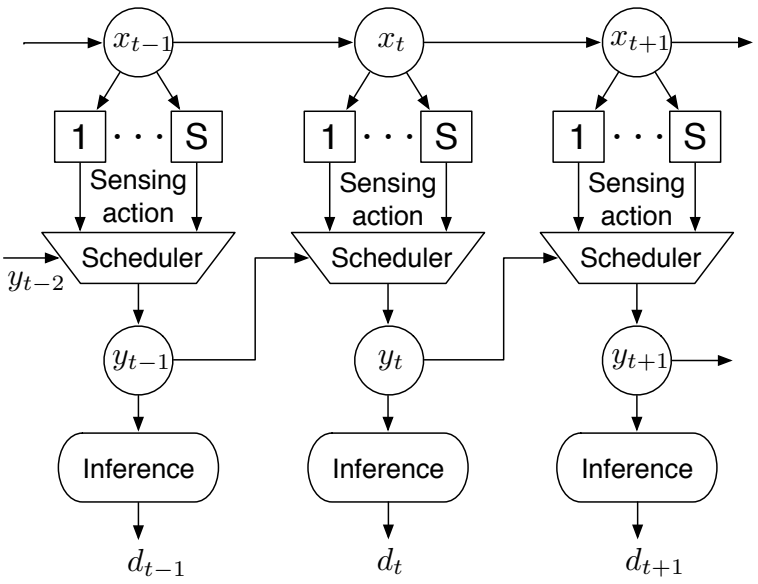


Figure 2.6: A standard illustration of an HMM augmented with multiple sensing actions and a closed-loop resource scheduler. A standard HMM would have only the states s_t and observations y_t from a single sensing action connected according to the typical HMM assumptions.

action is generated according to some function η_t:

$$s_t = \eta_t(I_t^s) \tag{2.4}$$

Sensing action s_t is then activated and a new observation y_t is acquired. As assumed, y_t is drawn from x_t which evolves as a Markov chain, independent of sensing actions and inference decisions. The information available for making the inference decision d_t consists of:

$$I_t^d = I_t^s \cup \{s_t, y_t\} \tag{2.5}$$

such that the inference engine is represented by function ϵ_t:

$$d_t = \epsilon_t(I_t^d) \tag{2.6}$$

Thus, the optimization problem boils down to finding the sequence of functions $\eta = \{\eta_1, \ldots, \eta_t, \ldots\}$ and $\epsilon = \{\epsilon_0, \ldots, \epsilon_t, \ldots, \}$. Solving this functional optimization problem requires stochastic control theory, which we review in the next section.

2.3 Review of POMDP, Methods, and Sensor Management

The formulation constructed in the previous section illustrates the fundamental problem for sensor management: sequential decisions are made over time, where each decision generates new observations that provide additional information [10]. When viewed as a decision process with temporal dynamics involved, sensor management has its foundations in Markov decision processes (MDP), and in particular, partially observable MDPs (POMDP) for managing noisy sensors.

In this section, we review the basic theory of POMDPs, and survey exact and approximate methods developed to solve this class of problems.

2.3.1 Partially observable Markov decision process

A Markov decision process (MDP) is a generalization of a Markov chain, where control actions are used to influence state transitions. A partially observable Markov decision process (POMDP) has the additional complexity that the underlying states are hidden; *noisy* observations are available for inferring the underlying state transitions.

To formally define a POMDP, we follow an exposition similar to the presentation in [10, Ch 2]. A POMDP is denoted by the tuple $\mathcal{M} = \langle \mathcal{X}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Y}, \mathcal{O}, \mathbf{b}, N \rangle$, consisting of a state space $\mathcal{X}$, an action space $\mathcal{A}$,

and an observation space $\mathcal{Y}$. We assume all of these spaces to be finite sets and operated in discrete time for N time steps. The extension to a continuous observation space $\mathcal{Y}$ is particularly relevant in signal processing applications, and will be discussed in Section 2.3.5. For notational convenience, we have assumed that all of the processes are stationary. Although the theory holds more generally (e.g., when the transition kernel and action spaces are time varying), the development is identical for finite horizons, and so for clarity of presentation, we drop the dependence on time. A description of the remaining components and assumptions follows:

- $\mathcal{T}$ is the state transition function, $\tau(x, a, x')$, which is the probability of transitioning to state x', given that the state started in x and action a was taken. It is assumed that initial prior distribution $\mathbf{b} \in \mathcal{B}(\mathcal{X})$ is given, where $\mathcal{B}$ denotes the probability simplex over $\mathcal{X}$.

- $\mathcal{R}$ is the reward function, where $r(x, a)$ is the scalar reward received for being in state x and taking action a.

- $\mathcal{O}$ is the observation function, $o(x, a, y)$, giving the probability of observing y when action a is taken in state x. It is assumed that at any given time t, the observations y_t are conditionally independent, depending only on the current state x_t and action a_t.

Given action sequence $\{a_0, \ldots, a_{N-1}\}$, the state evolves as a Markov process. In particular, the effect of taking action a_t in state x_t depends only on the current value of the state and not on the prior history of the state.

The *information flow* at time t is as follows: the world is in state $x_t \in \mathcal{X}$. Action $a_t \in \mathcal{A}$ is taken, and reward $r(x_t, a_t)$ is received. The state evolves to $x_{t+1} \in \mathcal{X}$ according to the transition kernel $\tau(x_t, a_t, x_{t+1})$, and an observation y_{t+1} is generated with probability $o(x_{t+1}, a_t, y_{t+1})$.

To determine the action a_t, the *information vector*, denoted as I_t, defined to be all of the actions taken up to the current time and the associated observations, is available for decision-making:

$$I_t \triangleq \{a_0, y_1, \ldots, a_{t-2}, y_{t-1}, a_{t-1}, y_t\} \tag{2.7}$$

An admissible policy at time t is defined as a sequence of mappings $M = \{\mu_0, \ldots, \mu_{N-1}\}$ with the property that:

$$\mu_t : I_t \to \mathcal{A} \tag{2.8}$$

An admissible policy M will generate sequences of well-defined random variables corresponding to the state, action, and observation trajectories for N time steps[1]. We define the total reward associated with these trajectories as:

$$R = r(x_N) + \sum_{t=0}^{N-1} r(x_t, a_t) \tag{2.9}$$

R is a well-defined random variable with mean value $\mathbb{E}_M[R]$, where the expectation is taken with respect to the states and observations as generated by M, starting with prior distribution $\mathbf{b}$. The objective for optimization is to find an admissible policy that maximizes the total expected reward:

$$\max_M \quad \mathbb{E}_M \left[r(x_N) + \sum_{t=0}^{N-1} r(x_t, a_t) \;\middle|\; \mathbf{b} \right] \tag{2.10}$$

[1]The state trajectory actually consists of $N+1$ time steps because action a_{N-1} induces one additional transition.

2.3.2 Information and belief states

Conceptually, the problem of partial observability can be converted into a completely observable Markov decision process. The benefit is that Bellman's principle of optimality follows immediately. The conversion is accomplished in two steps: 1) by defining a different state space that preserves the dynamics and information flow of the original problem, and 2) finding an equivalent reward function for this new space.

Instead of the original state $x_t \in \mathcal{X}$, take I_t, the information vector defined in (2.7) to be the new state. The information vector can be written as a one-step recursion:

$$I_{t+1} = I_t \cup \{a_t, y_{t+1}\} \tag{2.11}$$

which is a function of the state at the previous time step, the subsequent control action taken (which depends only on I_{t-1}), and current observation (which depends only on a_{t-1} and x_t). Note that the observation can be interpreted as a random disturbance in the evolution of the information state [11, Ch 5], and is independent of prior disturbances:

$$P(y_{t+1} \mid I_t, a_t) = P(y_{t+1} \mid I_t, a_t, y_0, \ldots, y_t) \tag{2.12}$$

which follows because prior observations are already contained in I_t, by definition.

The reward function for this new space must be of the form $\bar{r}(I, a)$, mapping the information state and control action to a scalar value that is equivalent to the original problem. To show that this can be done, we use the smoothing property of conditional expectations to write the objective func-

tion in (2.10) as:

$$\mathbb{E}_M\left[\sum_{t=0}^{N-1} r(x_t, a_t)\right] = \mathbb{E}_M\left[\sum_{t=0}^{N-1} \mathbb{E}_M\left[r(x_t, a_t) \mid I_t, a_t\right]\right] \tag{2.13}$$

$$= \mathbb{E}_M\left[\sum_{t=0}^{N-1} \bar{r}(I_t, a_t)\right] \tag{2.14}$$

where the expectations are taken with respect to policy M. Note that in this new problem, the structure of the policy has not changed (i.e., the mapping defined in (2.8) where $a_t = \mu_t(I_t)$ still holds).

Although formulating an equivalent MDP implies that all of the theoretical results for Markov decision problems apply directly to the partially observable problem, the resulting DP algorithm must be carried out for a state space with dimension that increases over time.

The trick to fundamentally reduce the problem size is to show that there is a *belief state* whose dimension does not grow with time, denoted as $\mathbf{b}$, that is sufficient to achieve the same optimal reward as using the information state for control. The appropriate belief state is the posterior probability $P(x_t \mid I_t)$, which is an $|\mathcal{X}|$-dimensional vector in the probability simplex $\mathcal{B}(\mathcal{X})$ and evolves recursively:

$$\mathbf{b}_t = P(x_t \mid I_t = \{I_{t-1}, a_{t-1}, y_t\}) \triangleq \Phi(\mathbf{b}_{t-1}, a_{t-1}, y_t) \tag{2.15}$$

At this point, we introduce a vector notation which will be convenient for subsequent discussion. In particular, for the finite space $\mathcal{X}$ with arbitrary elements x_i for $i = 1, \ldots, |\mathcal{X}|$, we can redefine $\mathcal{X} = \{1, \ldots, |\mathcal{X}|\}$, such that stating that index $i \in \mathcal{X}$ is equivalent to stating that $x_i \in \mathcal{X}$.

The significance of (2.15) is that when propagating posterior belief, the information state I_{t-1} can be replaced by the prior belief b_{t-1}, whose di-

mensionality does not grow with time. This simplification follows from the Markov assumptions, and the operator Φ which implements Bayes' rule, simplifies to the following: for each state $j \in \mathcal{X}$,

$$\Phi_j(\mathbf{b}, a, y) \triangleq \mathbf{b}_a^y(j) \tag{2.16}$$

$$= \frac{o(j, a, y)}{p(y|a, \mathbf{b})} \sum_{i \in \mathcal{X}} \tau(i, a, j)\mathbf{b}(i) \tag{2.17}$$

where $p(y|a, \mathbf{b}) = \sum_j o(j, a, y) \sum_i \tau(i, a, j)\mathbf{b}(i)$ is a normalizing factor.

Completing the conversion of the information-state MDP to an equivalent belief-state MDP requires showing that there is an equivalent reward function for the belief-state space (i.e., $\tilde{r}(\mathbf{b}, a)$). This follows immediately because

$$\bar{r}(I_t, a_t) = \mathbb{E}_M \left[r(x_t, a_t) \mid I_t, a_t \right] = \sum_{j \in \mathcal{X}} r(x_t = j, a_t)\mathbf{b}_t(j) = \tilde{r}(\mathbf{b}, a) \tag{2.18}$$

The procedure provided in this section for converting the partially observable problem into an information-state MDP and belief-state MDP is standard exposition and a major classical result in stochastic control theory [11, 12]. The implication for these conversions is 1) the ability to invoke standard dynamic programming techniques for finding the optimal policy, and 2) a fundamental reduction in the size of the search space.

2.3.3 Dynamic programming

Given that the belief state $\mathbf{b}$, evolving according to Φ, is sufficient for planning, the problem then is to construct a policy $M = \{\mu_0, \ldots, \mu_{N-1}\}$ where $\mu_t : \mathcal{B} \to \mathcal{A}$ maps belief states to actions. A policy M is evaluated by the value function $V_N^M : \mathcal{B} \to \mathbb{R}$, which is defined to be the expected reward for

implementing policy M, evaluated at some initial belief point $\mathbf{b}$

$$V_0^M(\mathbf{b}) = \mathbb{E}_M \left[\tilde{r}(\mathbf{b}_N) + \sum_{t=0}^{N-1} \tilde{r}\left(\mathbf{b}_t, \mu_t(\mathbf{b}_t)\right) \,\middle|\, \mathbf{b}_0 = \mathbf{b} \right] \tag{2.19}$$

The problem then is to find an admissible policy M^* such that

$$V_0^*(\mathbf{b}) = \max_M V_0^M(\mathbf{b}) \tag{2.20}$$

Such a policy M^* is said to be optimal. The optimal value function is characterized by the principle of optimality, which applies because the problem was shown to be an MDP. Starting from the last time step,

$$V_N^*(\mathbf{b}) = \max_{a \in \mathcal{A}} \tilde{r}(\mathbf{b}, a) \tag{2.21}$$

With V_N^* fully defined, the principle of optimality proceeds backwards, recursively. For $0 < n < N$:

$$V_{n-1}^*(\mathbf{b}) = \max_{a \in \mathcal{A}} \tilde{r}(\mathbf{b}, a) + \mathbb{E}\left[V_n^*(\mathbf{b}_a^y) \,|a, \mathbf{b}\right] \tag{2.22}$$

where the expectation is taken with respect to observation y (i.e., $p(y|a, \mathbf{b})$, which shows up in (2.16)).

It is straightforward to show that the optimal value functions are piecewise linear and convex (PWLC) functions [13]. This is a particularly powerful property since the piecewise linear property enables a *finite* representation of the value function, which is, in general, a point in an infinite-dimensional functional space due to the belief state being continuous. This finite number of linear segments are known as α-vectors, and the PWLC property can be derived using induction.

First, note that $\tilde{r}(\mathbf{b}, a)$ is a linear function of $\mathbf{b}$:

$$\tilde{r}(\mathbf{b}, a) = \sum_{j \in \mathcal{X}} r(x_t = j, a)\mathbf{b}(j) = \mathbf{r}_a^\top \mathbf{b} \tag{2.23}$$

where $\mathbf{r}_a$ is a column vector with element $\mathbf{r}_a(j) = r(j, a)$. For the induction step, assume that at time $n+1$ the set of α-vectors are given by Γ_{n+1}. The PWLC property implies that the value function can be represented as

$$V_{n+1}^*(\mathbf{b}) = \max_{\alpha \in \Gamma_{n+1}} \alpha^\top \mathbf{b} \tag{2.24}$$

Then, expanding the expectation in (2.22):

$$V_n^*(\mathbf{b}) = \max_{a \in \mathcal{A}} \mathbf{r}_a^\top \mathbf{b} + \sum_{y \in \mathcal{Y}} p(y|a, \mathbf{b}) \cdot V_{n+1}(\mathbf{b}_a^y) \tag{2.25}$$

$$= \max_{a \in \mathcal{A}} \mathbf{r}_a^\top \mathbf{b} + \sum_{y \in \mathcal{Y}} p(y|a, \mathbf{b}) \cdot \max_{\alpha \in \Gamma_{n+1}} \alpha^\top \mathbf{b}_a^y \tag{2.26}$$

$$= \max_{a \in \mathcal{A}} \mathbf{r}_a^\top \mathbf{b} + \sum_{y \in \mathcal{Y}} p(y|a, \mathbf{b}) \cdot \max_{\alpha \in \Gamma_{n+1}} \sum_{j \in \mathcal{X}} \alpha(j) \frac{o(j, a, y)}{p(y|a, \mathbf{b})} \sum_{i \in \mathcal{X}} \tau(i, a, j)\mathbf{b}(i) \tag{2.27}$$

$$= \max_{a \in \mathcal{A}} \mathbf{r}_a^\top \mathbf{b} + \sum_{y \in \mathcal{Y}} \max_{\alpha \in \Gamma_{n+1}} \sum_{i \in \mathcal{X}} \mathbf{b}(i) \sum_{j \in \mathcal{X}} \alpha(j) o(j, a, y) \tau(i, a, j) \tag{2.28}$$

$$= \max_{a \in \mathcal{A}} \mathbf{r}_a^\top \mathbf{b} + \sum_{y \in \mathcal{Y}} \max_{\alpha \in \Gamma_{n+1}} \sum_{i \in \mathcal{X}} \mathbf{b}(i) \beta_{a,y}^\alpha(i) \tag{2.29}$$

$$= \max_{a \in \mathcal{A}} \mathbf{r}_a^\top \mathbf{b} + \sum_{y \in \mathcal{Y}} \max_{\alpha \in \Gamma_{n+1}} \beta_{a,y}^{\alpha\top} \mathbf{b} \tag{2.30}$$

$$= \max_{a \in \mathcal{A}} \left(\mathbf{r}_a + \sum_{y \in \mathcal{Y}} \beta_{a,y}^{\alpha_b^*} \right)^\top \mathbf{b} \tag{2.31}$$

where

$$\alpha_b^* = \arg \max_{\alpha \in \Gamma_{n+1}} \beta_{a,y}^{\alpha\top} \mathbf{b} \tag{2.32}$$

$$\beta_{a,y}^{\alpha}(i) = \sum_{j \in \mathcal{X}} \alpha(j) o(j, a, y) \tau(i, a, j) \tag{2.33}$$

The number of alpha vectors in each Γ_n is finite with a maximum size of $|\mathcal{A}||\Gamma_{n+1}|^{|\mathcal{Y}|}$. Thus, because V_n^* in (2.31) has the same form (i.e., max over linear segments) as V_{n+1}^* in (2.24), piecewise linear convexity is preserved for all n.

The PWLC property makes policy search computationally feasible because although we are searching for a policy in a continuous (uncountably infinite) space MDP, there is a finite representation of the optimal value function at every iteration. Unfortunately, exact value iteration is only computationally feasible for small problems because the set of alpha vectors generated at each time step can grow exponentially in the worst case. This is because Γ_n must be generated for all $\mathbf{b}$ simultaneously. Exact value iteration algorithms do this by maintaining a minimal set of belief points which serve as a "witness" to the region for which a corresponding α-vector is maximal. The way in which these belief points and α-vectors are generated form the basis for the different types of algorithms [14, 15]. Open-source software for exact algorithms is available at [16].

2.3.4 Infinite horizons

Although we have presented the theory up to this point for finite-horizon problems, the computationally exact algorithms actually solve problems with discounted infinite horizons. This is because the optimal policy can be shown

to be time invariant under suitable conditions [11]. That is, $M^* = \{\mu, \mu \ldots\}$, assuming transition and observation probabilities are stationary and $\mathcal{A}_t = \mathcal{A}$ for all t. Practically, this reduces and simplifies the memory requirements for the representation of an optimal policy.

The only change in the problem formulation is reflected in the objective function. In particular, the search for an optimal policy is restricted to stationary policies, now denoted simply by μ, and is evaluated by the value function $V^\mu : \mathcal{B} \to \mathbb{R}$:

$$V^\mu(\mathbf{b}) = \mathbb{E}_\mu \left[\sum_{t=0}^{\infty} \gamma^t \cdot \sum_{x \in \mathcal{X}} r\left(x, \mu(\mathbf{b}_t)\right) \cdot \mathbf{b}_t(x) \;\middle|\; \mathbf{b}_0 = \mathbf{b} \right] \tag{2.34}$$

where $\gamma < 1$ is the discount factor.

The optimal value function V^* is the unique solution to the Bellman equation:

$$V^*(\mathbf{b}) = \max_{a \in \mathcal{A}} \; \mathbf{r}_a^\top \mathbf{b} + \gamma \cdot \mathbb{E}\left[V^*(\mathbf{b}_a^y) \;|a, \mathbf{b}\right] \tag{2.35}$$

This equation can be written as $V^* = HV^*$, where H is a contraction for $\gamma < 1$, from which existence and uniqueness of V^* follows from the contraction mapping theorem.

The optimal value function for an infinite horizon may not have the PWLC property. The issue is that in the limit, the number of alpha vectors may not be finite. There is a special class of policies, called finitely transient policies that are exactly PWLC [17], but the conditions are restrictive and this property does not hold for many practical and interesting problems.

Although the optimal value function for any infinite-horizon POMDP can be approximated arbitrarily well by a PWLC function, the same computa-

tional complexity that plagues finite horizon representations is exacerbated for infinite horizons, since the minimal set of α-vectors may grow unboundedly [17]. This has resulted in the use of approximate representations [18], as discussed in the next section.

2.3.5 Point-based methods

Motivations for solving large POMDPs have come mainly from the AI literature, with applications ranging from robot navigation to path planning to human-machine spoken dialog management systems [19]. Approximate methods developed to scale up and handle problems exhibiting hundreds and thousands of states include policies based on finite-state controller representations [20], compressed belief space [21], state space compression [22], and most productively in recent years, point-based methods [23, 24] which have laid the foundation for state-of-the-art POMDP solvers [25].

At their core, point-based solvers still perform value iteration, attempting to optimize the original POMDP problem. Efficiency is derived by approximating the value function with α-vectors at a finite set of belief points that do not grow exponentially over iterations. What differentiates the different point-based solvers is in how these belief points are maintained, which range from random initialization [26], depth-first search [27], and breadth-first search [28]. Computable lower and upper bounds are generally maintained to guide the maintenance of the set of belief points, and provide a confidence interval as to the optimality of the current value function.

The PWLC POMDP theory and methods are well established for finite state, observation, and action spaces. Extensions to continuous observation spaces (assuming finite state and action spaces) were presented in [29]. In-

terestingly, the optimal policy *induces* a finite partitioning of the observation space, which follows because observations are only relevant for deciding between a *finite* number of possible actions. Thus, by interpreting each region of the observation space as an aggregate observation, computational methods for solving problems with continuous observations are a straightforward extension of existing methods for finite observations.

Extensions to continuous state spaces are presented in [30], which is motivated by robot applications where the state space is modeled more naturally as a continuous space. The difficulty for value iteration methods is evaluating the expectation, which requires integration; tractability is gained by assuming all probability densities are Gaussian mixtures. Software packages for solving problems in continuous spaces are available online at [31].

2.3.6 Sensor management

The general POMDP solvers presented in the last section were developed mainly for robotics applications where a policy generally actuates something in the physical world in order to receive a reward. The context of decision processes used in this thesis corresponds more closely to a class of applications known as *sensor management* [32, 10] or *adaptive sensing* [33] – providing control in problems that are fundamentally about information discovery. Applications include active radar, sensor scheduling for target tracking, and data fusion from systems with multiple controllable sensing modalities.

Sensor-management problems have many of the same challenges as general POMDP problems, but do differ in some aspects. In particular, a POMDP is a more general decision process than what is required for sensor management because sensor selection "actions" typically do not affect or control

the dynamics of the physical state space (i.e., $\tau(s, a, s') = \tau(s, a', s')$ for all $a, a' \in \mathcal{A}$). The problem is still challenging for several reasons including 1) large state spaces (e.g., sleep management in sensor networks [34, 35]), 2) complicated physical dynamics to track (requiring particle filtering for belief updates), and 3) large [36] action spaces.

Computational methods specifically for sensor management have focused mainly on approximate methods, which aim to estimate the Q-function in a computationally efficient manner, as opposed to performing value iteration on the value function. Rewritten more suggestively,

$$Q(b, a) = r(b, a) + \gamma \cdot \mathbb{E}_a \left[V^*(b') \mid b \right] \tag{2.36}$$

we see that the difficulty in computing Q lies in evaluating the expected value-to-go (EVTG), which is the second term in (2.36). As the optimal value function V^* is ultimately responsible for providing a *ranking* of the possible sensing actions, the values themselves do not particularly matter, as long as the correct action for a given belief state is taken. As such, successful approximation techniques generally depend on the application at hand, and require some intuition about the future availability of information provided by the sensors in order to provide sensible ranking of actions. Methods to approximate the EVTG include replacing the EVTG with an information gain function, reinforcement learning, and policy rollout; see [10, Ch 5] for an overview of approximate methods used in sensor management.

Recently, point-based methods have been applied to schedule sensors in the context of energy-efficient tracking in sensor networks [37], and have been shown to outperform approximate methods based on surrogate value functions. As discussed in the previous section, advancements in point-based

methods are enabling problems that were previously too big to solve.

There is a body of sensor management literature that does not fit into the standard POMDP formulation, requiring extensions and generalizations. The standard POMDP assumes an average reward function, which is linear in belief. [38, 39] consider *non-linear* reward/cost functions, including the MMSE, minimax, and information measures as criterions. Constrained POMDPs are considered in [40, 41] and are motivated by hard resource constraints such as total-energy or time-to-completion constraints. This is a much harder problem, and Lagrangian relaxation and receding horizon control techniques have been proposed to handle these challenges.

The information flow found in sensor management problems shares a connection to a recent technique developed to manage the processing time in difficult computer vision problems, where non-myopic reinforcement learning is used to find a control policy for dynamic feature selection [42].

Finally, an approach called information-driven sensor management considers information measures as a surrogate reward function to be used in greedy myopic algorithms. [43] shows that the performance of a greedy algorithm is within 1/2 of optimal.

2.4 On Numerical Tractability

Given the capabilities and limitations of state-of-the-art POMDP solution techniques, in this thesis, we give up generality for tractability by limiting the scope of the problem formulation presented in Section 2.2. We focus on numerically tractable problems in order to establish the utility of developing a system-level tool that can be used to guide the design of energy-efficient sensing devices. Theoretical analysis and solution characterization for more

general variants of the problem formulated in Section 2.2 are left for future study.

The defining features that make a problem computationally tractable may not be immediately obvious to system designers not well-versed in the application of stochastic control theory. In this section, we explicitly describe the limited scope of the problem formulation, why they lead to tractability, and their implications.

There are five factors that influence the tractability of solving the problem formulated in Section 2.2; the time horizon, structure of the reward function, number of states, size of the observation space, and size of the action space all influence the technique used to find the optimal solution.

In this thesis, we consider *discounted infinite horizons*. The stochastic arrival of rare events implies that long (essentially infinite) time steps should be accounted for. Discounting is somewhat of a mismatch for the applications of interest because the energy consumed and penalty for an estimation error are typically computed as time averages; said another way, there is no reason why *earlier* rewards should be preferred. Although the theory for average-reward infinite horizons for completely observable MDPs is well-developed [11], partially observable problems are much more difficult to analyze [44]. Although a finite horizon eliminates the need for discounting, in general, the optimal policy for a finite-horizon problem is time-dependent, increasing the memory space required to compute an optimal policy.

We consider reward functions that at each time step, *depend only on the current state*. Discounting implies that the rewards are additive over time. This is a fundamental assumption for dynamic programming. Thus, we only consider Bayesian *filtering* tasks. Inference tasks requiring smoothing are not considered because the reward received at a particular time step would

require the value of future states. The implication is that the Viterbi decoder is not included in our analysis[2].

In expectation, reward functions are assumed to be *linear in belief*. The implication is that a broad class of objective functions, such as minimum mean squared error and information-based measures are excluded from our analysis. As standard analysis does not lead to structurally efficient characterizations [45], sophisticated techniques are required, as demonstrated by [39, 46].

We consider *state spaces of size greater than two*. Sequential binary detection is an important field specializing POMDP to binary states. Structural results such as the celebrated sequential probability ratio test are conceptually simple, but do not easily generalize. Furthermore, numerical approximation techniques that work for a two-dimensional belief simplex (such as uniform discretization), become computationally prohibitive for problems of even moderately greater size.

Finally, in this thesis, we assume that the *observation and action spaces are discrete*. The implications can be understood by considering the Bellman equation, (2.35). In particular, the size of the observation space affects the tractability of computing the expectation. In general, continuous observations will require numerical integration. The size of the action space determines how the maximization is computed; for typical applications, the number of sensing actions available on a device is finite and relatively small, compared to problems where the number of actions grows exponentially with some problem parameter (e.g., in [34], given N sensors in a network where

[2]It should be noted that the controlled Viterbi decoder can be shown to satisfy the principle of optimality, even with a non-causal reward function. This follows because the original Viterbi decoder is a shortest path problem. Unfortunately, this is computationally much harder to solve due to an augmented state space that has a mix of discrete and continuous states.

each sensor can either *sleep* or *sense*, the number of actions is 2^N).

2.5 Establishing the Principle of Optimality

The limitations on the time horizon and reward structure have implications for the objective function used for optimization. In particular, we assume the reward function is of the following form:

$$R = g(x_0, d_0) + \sum_{t=1}^{\infty} \gamma^t \cdot [g(x_t, d_t) - \lambda \cdot c(x_t, s_t)] \tag{2.37}$$

where $\lambda \geq 0$ is a Lagrange multiplier, balancing accurate inference with sensing costs. At each time step, a sensor s_t is utilized, incurring cost $c(x_t, s_t)$ in order to make an observation, which is used to make an inference decision d_t and receive reward $g(x_t, d_t)$. For the initial time step, we do not penalize for sensor usage, assuming that the inference decision d_0 is made using only prior knowledge. We assume that both the inference reward and energy-consumption costs are bounded (i.e., $|g(x, d)| < \infty$ and $|c(x, s)| < \infty$ for all $x \in X$, $d \in \mathcal{D}$, and $s \in \mathcal{S}$).

The sensors $\{s_t\}$ and decisions $\{d_t\}$ are sequences of well-defined random variables, generated according to policies η and ϵ, respectively. At the beginning of time t, the sensing action is chosen using all of the information available thus far, which is I_t^s defined in (2.3):

$$s_t = \eta(I_t^s) \tag{2.38}$$

After observation y_t is generated according to sensing action s_t, an inference

decision is made using $I_t^d = I_t^s \cup \{s_t, y_t\}$, also defined in (2.5):

$$d_t = \epsilon(I_t^d) \tag{2.39}$$

Thus, R is a well-defined random variable with mean $\mathbb{E}_{(\epsilon,\eta)}[R]$, where the expectation is induced by policies ϵ and η. The objective for optimization then is to find admissible policies that maximize the total expected reward:

$$\max_{\epsilon,\eta} \quad \mathbb{E}_{(\epsilon,\eta)}\left[g(x_0, d_0) + \sum_{t=1}^{\infty} \gamma^t \cdot (g(x_t, d_t) - \lambda \cdot c(x_t, s_t))\right] \tag{2.40}$$

At face value, this problem differs from a standard POMDP due to the *non-classical* flow of information. In particular, at any given time step, two decisions are made – a sensing decision at the beginning of the time step, and after an observation is drawn according to the chosen sensing action, an inference decision at the end of the epoch. In the standard POMDP setup, there is only a single action, which occurs before the state transition and observation is drawn.

To make the connection to the standard information flow, we borrow a technique from [47, Ch 3] to shift forward the boundary of what is defined to be an epoch, combining the inference decision at the current time step with the sensing decision from the next time step. This trick is made possible because the evolution of the hidden state x_t is not affected by any of the decisions that are made. Thus, defining $a_t = (d_t, s_{t+1})$, both decisions share the same information vector:

$$I_t = \{d_0, s_1, y_1, d_1, s_2, y_2, \ldots, d_{t-1}, s_t, y_t\} \tag{2.41}$$

$$= \{a_0, y_1, a_1, y_2, \ldots, a_{t-1}, y_t\} \tag{2.42}$$

and

$$a_t \triangleq \mu(I_t) = (\epsilon(I_t^d), \eta(I_{t+1}^s)) \tag{2.43}$$

where the information sets I_t^d and I_{t+1}^s can be generated using the information vector I_t (i.e., $I_t^d = I_t$, and $I_{t+1}^s = \{I_t \cup \epsilon(I_t)\}$).

Observe that the information vector defined as (2.42) has the same form as the standard information vector defined in (2.7). To finish the conversion to a completely observable MDP, we must find a reward function of the form $\bar{r}(I_t, a_t)$ that is equivalent to the original problem (2.40):

$$\mathbb{E}_{\epsilon,\eta}\left[g(x_0, d_0) + \sum_{t=1}^{\infty} \gamma^t \cdot (g(x_t, d_t) - \lambda \cdot c(x_t, s_t))\right] \tag{2.44}$$

$$= \mathbb{E}_{\mu}\left[\sum_{t=0}^{\infty} \gamma^t \cdot \mathbb{E}_{\mu}\left[g(x_t, d_t) - \gamma\lambda\mathbb{E}\left[c(x_{t+1}, s_{t+1}) \mid x_t, s_{t+1}\right] \mid I_t, a_t\right]\right] \tag{2.45}$$

$$= \mathbb{E}_{\mu}\left[\sum_{t=0}^{\infty} \gamma^t \cdot \mathbb{E}_{\mu}\left[g(x_t, d_t) - \lambda\tilde{c}(x_t, s_{t+1}) \mid I_t, a_t\right]\right] \tag{2.46}$$

$$= \mathbb{E}_{\mu}\left[\sum_{t=0}^{\infty} \gamma^t \cdot \bar{r}(I_t, a_t)\right] \tag{2.47}$$

where the second line follows from the law of iterated expectation and by grouping the expected sensor cost at the next time-step with the reward for the current inference decision.

Due to the standard Markov assumptions, it is straightforward to show that the posterior belief $P(x_t \mid I_t)$ is a sufficient statistic for the information state and evolves according to (2.15). In similar fashion as (2.18), the equivalent

reward function $\tilde{r}(\mathbf{b}_t, a_t)$ is defined as follows:

$$\bar{r}(I_t, a_t) = \mathbb{E}_\mu \left[g(x_t, d_t) - \lambda \tilde{c}(x_t, s_{t+1}) \mid I_t, a_t \right] \tag{2.48}$$

$$= \sum_{j \in \mathcal{X}} \left[g(x_t = j, d_t) - \lambda \tilde{c}(x_t = j, s_{t+1}) \right] \mathbf{b}_t(j) = \tilde{r}(\mathbf{b}_t, a_t) \tag{2.49}$$

where $a_t = (d_t, s_{t+1})$.

The non-classical information flow of the decision process considered in this thesis (i.e., scheduling, observation, followed by estimation) was first presented by Evans and Krishnamurthy in [48], where the problem is interpreted as a controlled hidden Markov model, as shown in Fig 2.7.

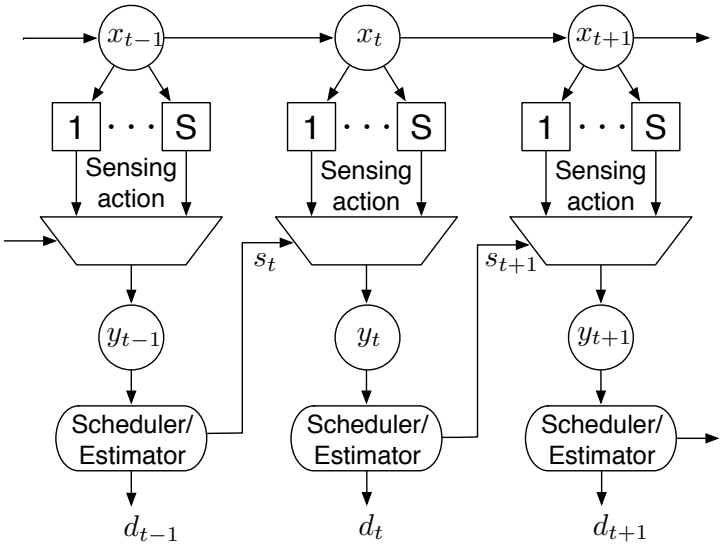


Figure 2.7: A controlled HMM, adapted from [48]. This figure is equivalent to Fig. 2.6. The inference decision at the current time step has been grouped together with the sensing decision to be used at the next time step.

Dynamic programming principles were derived from first principles for non-linear estimators (i.e., note the difference in notation between our inference decisions d_t and estimators $\hat{x}_t$ in Fig. 2.7) for finite horizons and continuous observations drive by a white noise process. In our formulation, we restrict the reward to be in linear form, assume a discounted infinite horizon, and

assume discrete observations; as noted before, this enables us to map the problem to a standard POMDP and subsequently use existing numerical solvers.

2.6 An Exact Direct Solver

A key observation from [48] was that the inference decision and scheduling problems are separable; intuitively, this can be seen in Fig. 2.7 where the inference decision d_t does not affect the future evolution of the belief state (i.e., $\mathbf{b}_{a_t}^{y_t} = \mathbf{b}_{s_t}^{y_t}$). Thus, the estimator (or inference decision) optimization can be done independently at each time step.

Mathematically, define $\mathbf{c}_s$ to be an $|\mathcal{X}|$-dimensional vector, with $\mathbf{c}_s(x) = \tilde{c}(x, s)$ for $x \in \mathcal{X}$. Then, writing the value iteration equation starting from (2.35) for iteration n:

$$V_n(\mathbf{b}) = \max_{a \in \mathcal{A}} \quad \mathbf{r}_a^\top \mathbf{b} + \gamma \mathbb{E}\left[V_{n-1}(\mathbf{b}_a^y) \mid a, \mathbf{b}\right] \tag{2.50}$$

$$= \max_{(d,s) \in \mathcal{A}} \quad \bar{\mathbf{r}}_d^\top \mathbf{b} - \lambda \mathbf{c}_s^\top \mathbf{b} + \gamma \mathbb{E}[V_{n-1}(\mathbf{b}_a^y) \mid a = (d, s), \mathbf{b}] \tag{2.51}$$

$$= \max_{d \in \mathcal{D}} \left\{\bar{\mathbf{r}}_d^\top \mathbf{b}\right\} + \max_{s \in \mathcal{S}} \left\{-\lambda \mathbf{c}_s^\top \mathbf{b} + \gamma \mathbb{E}[V_{n-1}(\mathbf{b}_s^y) \mid s, \mathbf{b}]\right\} \tag{2.52}$$

where the maximizations in the last line separate because the belief update does not depend on inference decision d (i.e., $\mathbf{b}_a^y = \mathbf{b}_s^y$). This is exactly why the scheduling/estimation problem is separable. The first max in (2.52) is the inference-decision optimization which occurs independently at each time step, and the second max represents the value-iteration step if one were to consider the following *auxiliary* POMDP: $\langle \mathcal{X}, \mathcal{S}, \mathcal{T}, -\lambda \cdot \mathcal{C}, \mathcal{Y}, \mathcal{O}, \gamma \rangle$. Using any POMDP solver, make one value iteration on the auxiliary problem, generating some minimum set of α-vectors, and denote this set as $\Gamma_n^\star$. This

step is faster than an iteration of the original problem due to the reduced size of the action space.

In the following, we present an exact algorithm for completing the value-iteration step for the original POMDP, which consists of "fusing" the inference-decision rewards into the auxiliary α-vectors. Define $\Gamma_{\mathcal{R}} = \{\mathbf{r}_d : d \in \mathcal{D}\}$, where $\mathbf{r}_d(x) = r(x, d)$ for $x \in \mathcal{X}$. Continuing from (2.52),

$$V_n(\mathbf{b}) = \max_{\phi \in \Gamma_{\mathcal{R}}} \phi^\top \mathbf{b} + \max_{\alpha \in \Gamma_n^\star} \alpha^\top \mathbf{b} \tag{2.53}$$

$$= \max_{\phi \in \Gamma_{\mathcal{R}},\, \alpha \in \Gamma_n^\star} (\phi + \alpha)^\top \mathbf{b} \tag{2.54}$$

$$= \max_{\beta \in \bar{\Gamma}_n} \beta^\top \mathbf{b} \tag{2.55}$$

where

$$\bar{\Gamma}_n = \Gamma_{\mathcal{R}} \oplus \Gamma_n^\star \triangleq \{\phi + \alpha : \phi \in \Gamma_{\mathcal{R}},\ \alpha \in \Gamma_n^\star\} \tag{2.56}$$

and $\oplus$ is known as the cross-sum operator. Assuming an exact method was used to generate $\Gamma_n^\star$, one must *enumerate* all of the possible α-vectors in (2.56) to represent the value function at iteration n. Subsequently, finding a minimum set of α-vectors requires a pruning step, resulting in the final set of α-vectors, $\Gamma_n = \text{Prune}(\bar{\Gamma}_n)$. In its simplest form, Prune systematically checks each vector in $\bar{\Gamma}_n$ to see if it dominates at some belief state, which serves as a witness that the vector is in the minimum set. The existence of

such a point can be determined by solving a linear program (LP):

$$\begin{aligned} & \max_{b \in \mathcal{B}} \quad \delta \\ \text{s.t.} \quad & \hat{\gamma}^\top \mathbf{b} \geq \gamma^\top \mathbf{b} + \delta \quad \forall\, \gamma \neq \hat{\gamma} \in \bar{\Gamma}_n \\ & \sum_{x=1}^{\mathcal{X}} \mathbf{b}(x) = 1 \\ & \mathbf{b}(x) \geq 0 \quad \forall\, x \in \mathcal{X} \end{aligned}$$

If the program does not return $\delta \geq 0$, then there is no belief point for which $\hat{\gamma}$ dominates, and so $\hat{\gamma}$ is removed. Once this procedure is completed for every vector in $\bar{\Gamma}_n$, Γ_n will be the minimum set. At worst, Prune($\bar{\Gamma}_n$) must solve $|\bar{\Gamma}_n|$ LPs. In [49], this direct exact algorithm was compared to running an exact algorithm which enumerates all possible combinations of sensing actions and inference decisions, demonstrating a speedup of a factor of $|\mathcal{X}|$.

2.7 Efficient Approximate Solution using Point-based Solvers

As discussed in Section 2.3.4, exact algorithms have been shown to be tractable only for small problems. In this section, we demonstrate how to map the problem into standard form, for which existing numerical point-based solvers can be used. For problems of practical size, we have found that the speedup from point-based approximations compensates for the inefficiency discussed in the previous section.

To review, the components of an discounted infinite horizon PWLC POMDP consist of $\mathcal{M} = \langle \mathcal{X}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Y}, \mathcal{O}, \gamma \rangle$. As described in the previous section, the action space $\mathcal{A}$ is the tuple $\mathcal{D} \times \mathcal{S}$. One way to represent this is

to enumerate all possible combinations of inference decisions and sensing actions. For example, assume $\mathcal{D} = \{absent,\ present\}$ and $\mathcal{S} = \{sense,\ sleep\}$. Then,

$$\begin{aligned}\mathcal{A} &= \{a_1, a_2, a_3, a_4\} \\ &= \{(absent,\ sense), (absent,\ sleep), (present,\ sense), (present,\ sleep)\}\end{aligned}$$

The implication is that the optimal PWLC POMDP policy that is generated by a numerical solver, denoted by μ^*, maps belief $\mathbf{b}$ to $\mathcal{A}$, and thus at time t, jointly specifies the inference decision d_t and next sensing action s_{t+1}. The process is illustrated in Fig. 2.8.

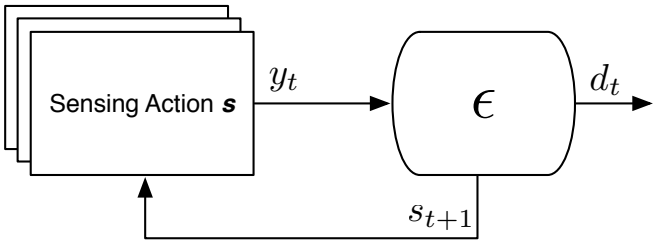


Figure 2.8: The problem of optimally scheduling sensing actions and making inference decisions can be cast as an optimization problem to find policy ϵ that maps a belief state to actions in $\mathcal{A}$, which jointly specifies the inference decision d_t, and sensing action s_{t+1}.

The remaining components of the standard PWLC POMDP are as follows:

- $\mathcal{X}$, the state space, was defined in Section 2.2.2 as the flattened hierarchical Markov chain that accounts for event arrivals
- $\mathcal{T}$, the transition function does not depend on the sensing action or inference decision. Thus,

$$\tau(x, a, x') = \Pr(X_{t+1} = x' | X_t = x) \quad \forall a \in \mathcal{A} \tag{2.57}$$

and satisfies the Markov property, as defined in (2.1).

- $\mathcal{R}$, the reward function is defined according to (2.48). For action $a \in \mathcal{A}$ defined to be $a = (d, s)$:

$$r(x, a) = g(x, d) - \lambda \cdot \tilde{c}(x, s) \tag{2.58}$$

- $\mathcal{Y}$ is the observation space. For action $a \in \mathcal{A}$ defined to be $a = (d, s)$, the observation function $\mathcal{O} : \mathcal{X} \times \mathcal{A} \times \mathcal{Y} \to [0, 1]$ is defined to be:

$$o(x, a, y) = \bar{o}(x, s, y) \tag{2.59}$$

  where $\bar{o}$ is defined in (2.2), and satisfies the conditional independence assumption. Note, the observation functions are independent of the inference decision d.

- $\gamma < 1$ is the discount factor as denoted in (2.37).

2.8 An Improved Lower Bound

In this section, we exploit special properties of the problem structure to derive a lower bound on the optimal value function, which is an improvement over the bound that is currently used in state-of-the-art point-based solvers for initializing the value function. We present a computational point-based method to compute the improved lower bound.

2.8.1 Motivation

In the problems of interest, where event arrivals are augmented to the state space, value iteration can take a long time to converge. This is due to the

slow mixing rate of the underlying Markov chain which arises because, as discussed in Section 2.2.2, we use the Markov chain to model event arrivals which can occur at a relatively slow time scale, especially when events are rare.

There are two problem characteristics that we exploit to derive this improved lower bound:

1. the separability property described in Section 2.6, which arises because inference decisions have no influence over the evolving belief state

2. the fact that sensing actions have no influence over the underlying state transitions

2.8.2 Assumptions

The first assumption we make is that "sleeping" is a valid sensing action. Typically, in energy-constrained applications, low-power modes are utilized to reduce energy consumption and turn off resources; thus, no observations can be generated when one chooses to sleep. A second assumption we make is that sleeping is the lowest resource-consuming sensing action. That is, $c(x, (d, \text{sleep})) \leq c(x, a)$ for all $x \in \mathcal{X}$, $d \in \mathcal{D}$, and $a \in \mathcal{A}$.

2.8.3 A standard lower bound: blind strategies

A blind strategy is used commonly in point-based solvers to form a lower bound to the optimal value function. A blind strategy executes the same action, regardless of belief and observations. The value function of a blind strategy executing action a, $Q_{blind}(\mathbf{b}, a)$, can be shown to be linear in belief

and computed in closed-form [50]:

$$Q_{blind}(\mathbf{b}, a) = \alpha_a^\top \mathbf{b}, \text{ with } \alpha_a = \left(I - \gamma\, \mathbf{T}^\top\right)^{-1} \mathbf{r}_a \tag{2.60}$$

where I is the $|\mathcal{X}| \times |\mathcal{X}|$ identity matrix. For a given belief state, choosing the *best* blind strategy can be constructed as:

$$V_{BS}(\mathbf{b}) = \max_{a \in \mathcal{A}} \; Q_{blind}(\mathbf{b}, a) \tag{2.61}$$

Note that the structure of the reward function in (2.37) gives us the following result for blind strategies:

Lemma 1

$$V_{BS}(\mathbf{b}) = \max_{a \in \mathcal{A}_{sleep}} \; Q_{blind}(\mathbf{b}, a) \tag{2.62}$$

where $A_{sleep} = \{(d, s) \in \mathcal{A} : s = sleep\}$.

Proof Since observations are ignored, the selected sensing action cannot affect inference performance. As sleeping is the lowest resource consumer, it will always be a part of the best blind strategy. Thus, the maximization in (2.61) can be limited to $\mathcal{A}_{sleep}$. ∎

2.8.4 An improved bound: a sleep strategy

By fixing a single action, a blind strategy ignores the fact that the inference decision is solved independently at each time step, depending only on the current belief state; this follows from the separability property. Thus, an improved base policy is what we call the *sleep strategy*. At every decision epoch:

1. use the current belief $\mathbf{b}$, to determine the optimal inference decision, $d = \arg\max_{d \in \mathcal{D}} \ \mathbf{r}_d^\top \mathbf{b}$

2. choose $s =$ sleep as the next sensing action

Thus the proposed sleep strategy, with value function $V_{SS}(\mathbf{b})$, differs from the blind-strategies policy in that the inference decision is belief-dependent. The following result proves that the sleep-strategy base policy improves upon the blind-strategies base policy:

Theorem 1 $V^*(\mathbf{b}) \geq V_{SS}(\mathbf{b}) \geq V_{BS}(\mathbf{b}) \quad \forall\, \mathbf{b} \in \mathcal{B}$.

Proof The first (leftmost) inequality is clear because both the sleep and blind strategies ignore observations. The following proof for the second inequality is useful for motivating the computational method presented next. Defining $\mathbf{P} \triangleq (I - \gamma\, \mathbf{T})^{-1}$:

$$V_{BS}(\mathbf{b}) = \max_{a \in \mathcal{A}_{sleep}} \mathbf{r}_a^\top \left(I - \gamma\, \mathbf{T}^\top\right)^{-1} \mathbf{b} \tag{2.63}$$

$$= \max_{a \in \mathcal{A}_{sleep}} \left(\mathbf{r}_d^\top - \lambda\, \mathbf{c}_{sleep}^\top\right) \mathbf{P}^\top \mathbf{b} \tag{2.64}$$

$$= -\lambda\, \mathbf{c}_{sleep}^\top \mathbf{P}^\top \mathbf{b} + \max_{d \in \mathcal{D}} \left\{\mathbf{r}_d^\top \mathbf{P}^\top \mathbf{b}\right\} \tag{2.65}$$

$$= -\lambda\, \mathbf{c}_{sleep}^\top \mathbf{P}^\top \mathbf{b} + \max_{d \in \mathcal{D}} \left\{\sum_{t=0}^{\infty} \gamma^t\, \mathbf{r}_d^\top \left(\mathbf{T}^t \mathbf{b}\right)\right\} \tag{2.66}$$

$$\leq -\lambda\, \mathbf{c}_{sleep}^\top \mathbf{P}^\top \mathbf{b} + \sum_{t=0}^{\infty} \gamma^t \max_{d \in \mathcal{D}} \left\{\mathbf{r}_d^\top \left(\mathbf{T}^t \mathbf{b}\right)\right\} \tag{2.67}$$

$$= V_{SS}(\mathbf{b}) \tag{2.68}$$

where the first line follows from Lemma 1, and (2.66) follows because $\mathbf{r}_d^\top \mathbf{P}^\top \mathbf{b}$ is the closed-form expression for the infinite discounted value of deciding d at every time step, starting at belief $\mathbf{b}$. The inequality follows because the max of a sum is always less than or equal to the sum of the max. (2.68) follows

because the value achieved in the infinite sum in (2.67) is exactly that of the optimal inference decision when sleeping is the sensing action that is always chosen – this is exactly the sleep strategy. ∎

2.8.5 Computing the Value of the Sleep Strategy

The summation in (2.68) is an infinite sum, but can be computed exactly for any belief $\mathbf{b}$. This follows because the belief trajectory $\mathbf{T}^t\mathbf{b}$ is deterministic over time, and converges to the equilibrium distribution, $\mathbf{b}_\pi$.

Define $B_\pi = \{\mathbf{b} \in \mathcal{B} : \arg\max_{d\in\mathcal{D}} \ \mathbf{r}_d^\top \mathbf{T}^t\mathbf{b} = d_\pi \ \forall \ t \geq 0\}$, with $d_\pi = \arg\max \ \mathbf{r}_d^\top \mathbf{b}_\pi$. Then, let N be the number of steps until $\mathbf{b}$ enters B_π, which is guaranteed to be finite since $\mathbf{b} \to \mathbf{b}_\pi$ at a geometric rate [51]. Then, for any initial belief $\mathbf{b}$, the infinite sum in (2.68) can be computed exactly:

$$\sum_{t=0}^{\infty} \gamma^t \cdot \max_{d\in\mathcal{D}} \ \mathbf{r}_d^\top \mathbf{T}^t\mathbf{b} = \sum_{t=0}^{\infty} \gamma^t \mathbf{r}_{d_t^*}^\top \mathbf{T}^t\mathbf{b} \tag{2.69}$$

$$= \sum_{t=0}^{N-1} \gamma^t \mathbf{r}_{d_t^*}^\top \mathbf{T}^t\mathbf{b} + \gamma^N \mathbf{r}_{d_\pi}^\top \mathbf{P}^\top \mathbf{T}^N\mathbf{b} \tag{2.70}$$

where d_t^* is the optimal inference decision at time t. The partial sum from N to ∞ is replaced by the value of the blind strategy (2.60), because once $\mathbf{b}$ enters B_π, the two strategies are equivalent.

A point-based algorithm to compute a lower bound on $V_{SS}(\mathbf{b}) = \max_{\alpha\in\Gamma_{SS}} \alpha^\top\mathbf{b}$ is to sample a set of belief points B_{SS} and compute the derivative (i.e., α-vector) at each point $\mathbf{b} \in B_{SS}$:

$$\alpha_b = \sum_{t=0}^{N_b-1} \gamma^t \left(\mathbf{T}^\top\right)^t \mathbf{r}_{d_t^*} + \gamma^{N_b} (\mathbf{T}^\top)^{N_b} \mathbf{P}\mathbf{r}_{d_\pi} - \lambda \, \mathbf{P}\mathbf{c}_{sleep} \tag{2.71}$$

where N_b is the number of steps to enter into B_π. Denote the resulting set of

α-vectors as Γ_{SS}, where $|\Gamma_{SS}| \leq |B_{SS}| + 1$, where strict inequality may occur due to duplicated α-vectors.

2.8.6 Discussion

The value of the sleep strategy can be computed offline, and is independent of the observation models. The sleep strategy is the intuitive thing to do when energy constraints are stringent and the device is forced to sleep. Although computing the value of the sleep strategy is complicated by the fact that optimal inference decision depends on the evolving belief, we have shown that this computation is in fact tractable because control actions do not influence state transitions.

Finally, from (2.71), the improved lower bound is linear in λ, which is the Lagrange multiplier controlling the trade-off between inference performance and resource utilization. The implication is that if resource availability changes, the partial sum in (2.71) does not need to be recomputed.

CHAPTER 3

ENERGY-EFFICIENT SENSING

3.1 Application Driver: Acoustic Wildlife Monitoring

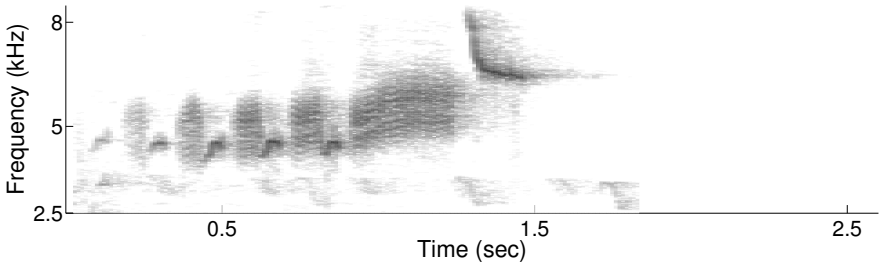
The particular application we consider is the acoustic monitoring of the golden-cheeked warbler (GCW), which is a federally listed endangered species of bird. Automated long-term monitoring has important implications for conservation, including 1) accurate population estimation, which is critical for assessing the status of the endangered species, and 2) providing scientists with data to identify evolutionary changes in the acoustic call structure [52]. This application is highly energy-constrained since continual monitoring on a single set of batteries is needed for an entire breeding season, which lasts for up to 4 months.

In this thesis, our goal is to enable sophisticated sensing and processing at considerably reduced energy consumption on a *single* sensor node. Cooperation with other nodes to increase acoustic coverage is considered as complementary work and not considered here.

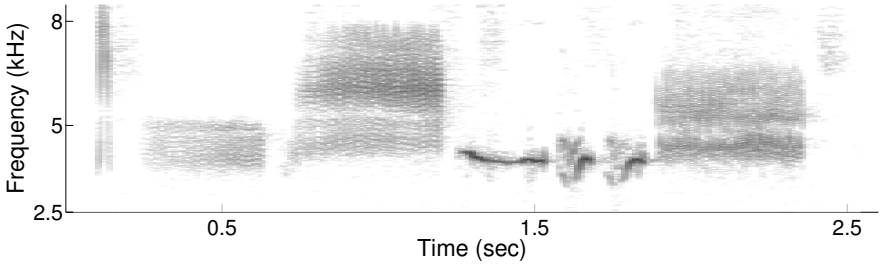
3.2 Acoustic Model

Male warblers exhibit acoustic activity early in the breeding season to attract female mates, and later in the season to defend their territory. The time-frequency structure of the two different types of acoustic calls is illustrated in

Fig. 3.1. Depending on the type, calls last anywhere from 1.5 to 2.5 seconds [53].



(a) Type A



(b) Type B

Figure 3.1: Spectrogram of the Type A and Type B calls.

Fig. 3.2 shows the hierarchical temporal structure of the bird activity over the course of five hours during the day[1]. In particular, a single bird calls at a steady rate of about once every 10-12 seconds, and activity persists for relatively long periods of time, on the order of 15-45 minutes at a time. Furthermore, physical locality dictates that when the bird is not in the vicinity of the recording device, there is good probability that it will be away for a while.

The rich temporal structure at multiple time scales creates for an interesting inference problem and opportunities to save device energy.

Since the focus of this thesis is not to build the most accurate bird classification algorithm, we make a few simplifying assumptions, keeping the narrative

[1]The recording was made by Wendy Leonard from the San Antonio Parks and Recreation Natural Areas, and Dr. Rama Ratnam from UT San Antonio on April 20, 2010.

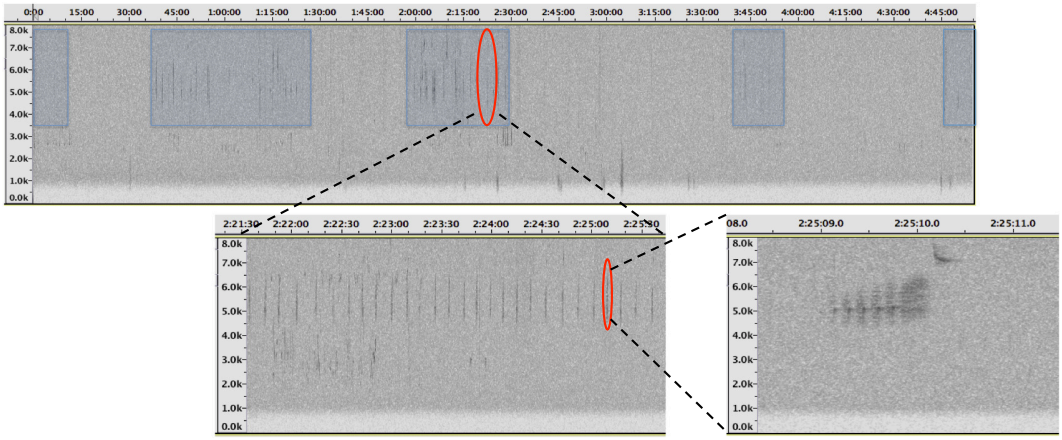


Figure 3.2: The temporal activity of GCW males over the course of a day. The highlighted sections in the top plot signify when the bird is present over a 5-hour recording session; progressively zooming in on a particular point in time uncovers structure at multiple time scales, ranging from hours to minutes to seconds.

relatively straightforward at the expense of monitoring performance:

1. Extract features based on a fixed block size of 500ms. The implication is that we ignore the fine-scale temporal structure, which would be on the order of 100ms, of the call signature shown in Fig. 3.1.

2. Assume that the call is a colored Gaussian signal with known covariance K in WGN with fixed noise power σ_n^2. The implication is that the most complicated feature we extract is bandpass-filtered energy, which approximates the estimator-correlator [54].

3. We only consider type A calls. Without fine-scale temporal patterns, it becomes harder to distinguish between the two call types. Inferring about type B calls would only double the state space size and increase the amount of learning that must be done.

4. Acoustic frames are conditionally independent given the state. This is more of an implicit assumption we make to leverage the HMM and POMDP theory.

3.2.1 A Simple Feature

We present a naive signal model that results in simple processing. In particular, we ignore the frequency structure of the colored gaussian signal. That is, we assume that when the bird is present, the acoustic data is modeled as a mean-zero, white Gaussian process with variance $|K| + \sigma_n^2$. When the bird is absent, the acoustic data is also white Gaussian noise with variance σ_n^2.

Under this signal model, the total energy, defined as:

$$T_1 \triangleq \sum_{i=1}^{N} y_i^2 \tag{3.1}$$

is a sufficient statistic for optimal detection. We denote the feature of this simple signal model as T_1. Evaluating the sample energy over the block of data consists of a single multiply-accumulate instruction per datum and can be easily performed within a hardware interrupt routine.

3.2.2 A More Accurate Feature

For a feature that exploits the known covariance structure of the bird call, we consider the *estimator-correlator* [54] for estimating a colored gaussian signal in WGN. From prior knowledge, we know that GCW bird calls are contained in the 4.5 to 7.5 kHz frequency range [53]. Thus, we approximate the estimator-correlator using a bandpass-energy detector. That is:

$$T_2 \triangleq \sum_{i=1}^{N} z_i^2 \tag{3.2}$$

where z_i is the output of the block of data filtered through an appropriate bandpass filter. Running a bandpass filter consists of basic DSP operations, and can be processor intensive depending on the filter order.

3.2.3 GCW Acoustic Data Trace

To validate the assumptions of our signal model, we plot the *detection* performance of the two signal models tested on five hours of the continuous data shown in Fig. 3.2. The optimal detectors are in the form of a likelihood ratio test:

$$T_i \underset{<}{\overset{>}{=}} \tau_i \tag{3.3}$$

where τ_i is a threshold that is varied to trace out the receiver operating characteristic (ROC) curve for detector i. Fig 3.3 plots the ROC curves of the two signal models *individually*, which plot true detection rate vs. false alarm rate.

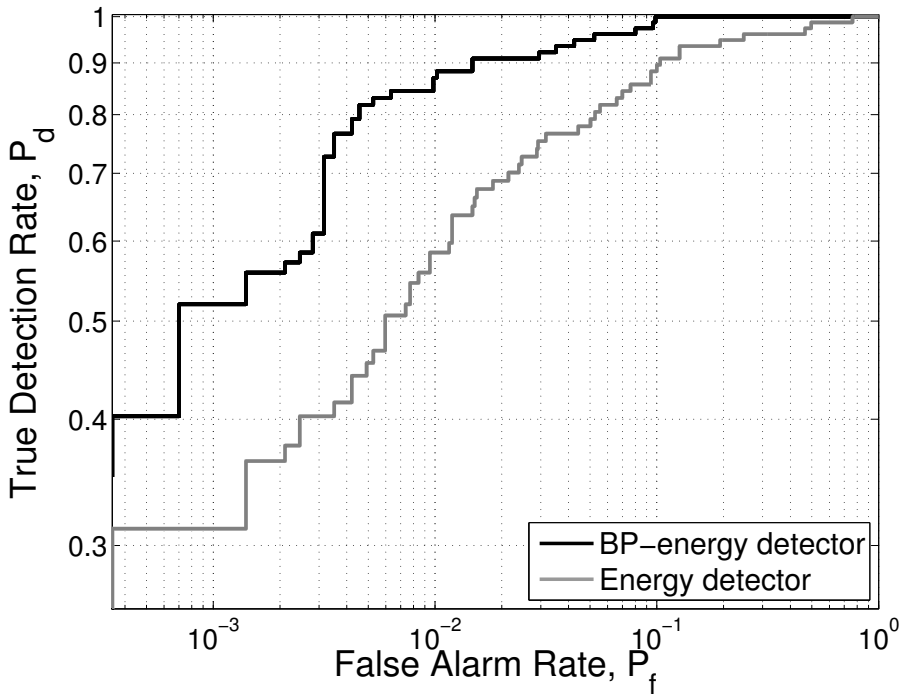


Figure 3.3: Detection performance for each signal model, run on 5 hours of continuously recorded data. As expected, the accurate signal model is uniformly better than the simple signal model.

3.3 CheetahCub Testbed

To understand how data acquisition and processing translate to energy consumption, the CheetahCub testbed is a platform we built based on the Texas Instruments (TI) MSP430 microcontroller, one of the lowest-power microcontrollers available on the market. Using commercially available components, the testbed has allowed us to explore the energy scalability of cheap commercial low-power embedded devices. The name is derived from the fact that cheetahs are the world's fastest land animals, and yet they rest for most of the day, which forms the intuition for achieving high performance with energy efficiency. The MSP430 is not particularly fast (up to 25MHz), and so it represents a baby version of more powerful embedded devices that are available on the market.

Table 3.1 lists key components and peripherals used in our testbed. The last component, a real-time energy monitor, would be a useful feature for closed-loop management, if the power consumption of the monitoring unit itself was negligible. Unfortunately, this is a non-trivial requirement, as the energy monitor itself should consume microwatts of power. Relevant commercially available energy monitors are designed for mobile phones, whose energy consumption is a couple orders of magnitude higher than the envisioned applications. [55] identified this paradox and recently proposed a clever simple, "energy-free" solution that exploits the linear relationship between load current and switch frequency found in certain switching regulators.

The testbed is shown in Fig. 3.4 and features a low-power MCU, controllable linear regulator, on-board microphone for acoustic sensing, microSD card for data archiving, and JTAG support for full-access debugging on the microcontroller.

Table 3.1: Hardware Component Description

Component	Description
Low-power MCU	Main board processor
Hardware multiplier	Efficient data processing
32kHz crystal	Low-power, high precision timing
Mic & preamp	Acoustic sensing
ADC	Samples analog signal
DMA	Autonomous, low-power data acquisition
Controllable regulator	Supply-voltage scaling between stages
microSD card	Data archiving
Energy monitor	Real-time energy monitoring (not available)

3.3.1 MSP430-based Design

The Texas Instruments MSP430F5438A is a low-power microcontroller with peripherals including a 12-bit ADC, direct memory access (DMA), and 32-bit hardware multiplier. A processor in the 5xx family was chosen for two reasons: 1) it supports higher clock frequencies (up to 25 MHz, as opposed to 16 MHz in other families), and 2) it has an integrated power management module (PMM), which allows dynamic scaling of the core voltage, and is also useful for safeguarding dynamic supply-voltage scaling.

The TI TPS780 is a low-drop-out regulator (LDO) which can be controlled to output either 2.2V or 3.3V; this output is used to power the MCU. As shown in Fig. 3.5, the control bit is provided by the MCU directly, resulting in closed-loop operation. The core voltage is also controllable, and is set in software in conjunction with the main clock frequency.

An electret microphone and pre-amplifier circuit based on the TI TLV2760, a low-power op-amp, is used to collect acoustic data. Shutdown capability (on and off in less than 5μs) is included in case continuous data collection is not required.

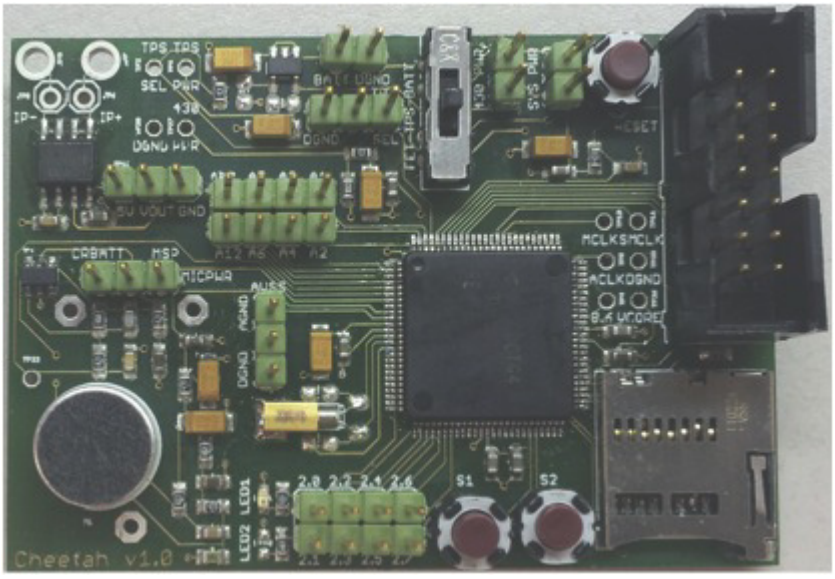


Figure 3.4: CheetahCub Testbed: an MSP430F5438A with a controllable LDO for DVFS and onboard acoustic sensor.

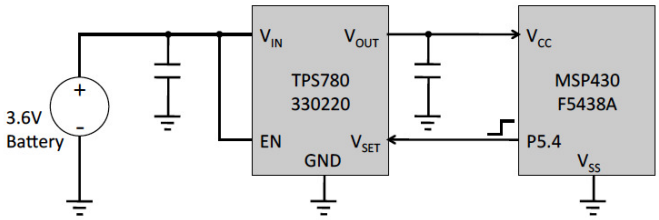


Figure 3.5: Closed-loop control of the MCU supply voltage.

3.3.2 Performance & Energy Scalability

Table 3.2 shows the average current consumed for the two processing tasks and breaks down where in the processing chain the power is consumed. The peripherals for the simple model consume less current because they are run at 2.2V. The average power for both scenarios is calculated by multiplying the average current by the power supply voltage, 3.3V.

Table 3.2: Average Current Consumption of the Two Signal Models

	Simple Model (μA)	**Accurate Model (μA)**
Microphone	190	200
Pre-amp	230	260
ADC	140	170
DMA	80	160
LRT	80 [18cps]	7600 [1330cps]
Total	**720**	**8400**

3.3.3 Design Efforts

As described, dynamic voltage and frequency scaling was projected to decrease the device-energy consumed by the lower-complexity energy detector, which runs for significantly more time than the higher-complexity processing algorithm. The baseline energy detector consumed 850μA average current, or 2.8mW of average power at 3.3V.

Operating V_{cc} at 2.2V reduced average current consumption down to 720μA, but the use of a linear regulator resulted in 66% efficiency. Thus, the average power consumed by the energy detector was 2.4mW instead of 1.6mW. At such low current consumption, switching regulators are not much more efficient and the added complexity requires a more in-depth study to determine if the potential energy savings warrants the extra engineering effort.

Furthermore, implementing dynamic voltage and frequency scaling requires external hardware and firmware safeguards to ensure processor stability during transition times. As illustrated in Fig. 3.6, this required significant engineering efforts for what was projected to be an additional 2× in energy savings, but instead ended up providing only 1.2× in additional savings.

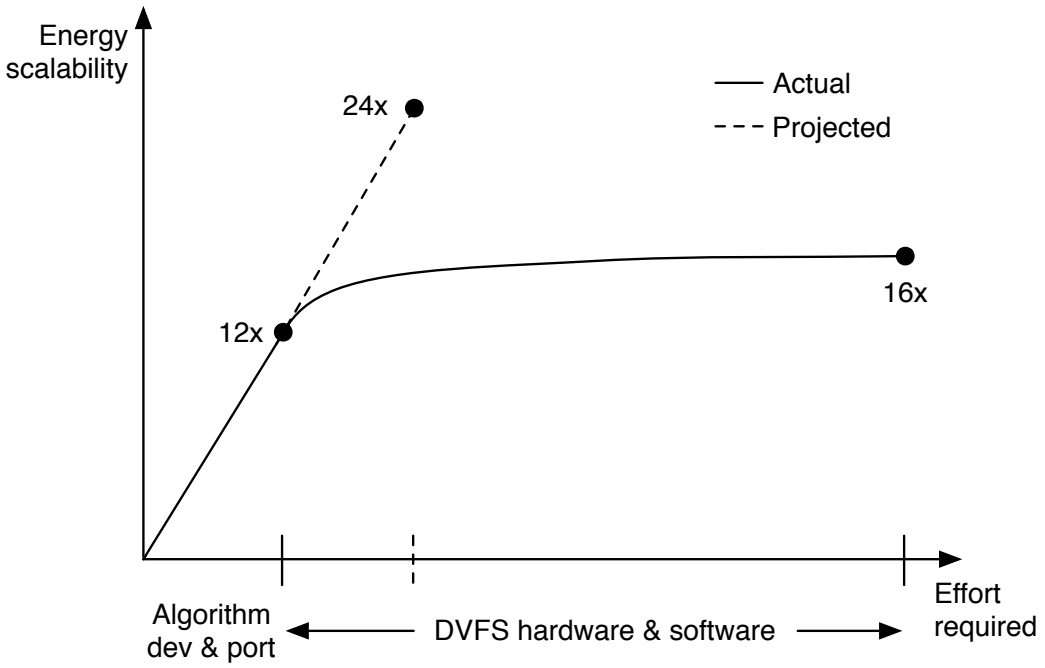


Figure 3.6: Illustrating the projected and actual energy scalability achieved with the development of the CheetahCub testbed, as a function of the required engineering effort.

3.4 EFM Testbed

While the TI MSP430 is optimized for sleeping, there is an emerging class of processors that are aiming to deliver high performance at greater energy efficiency. In particular, the ARM Cortex-M series is a family of 32-bit microcontrollers ranging from the highest-end processor containing a floating-point unit, to the lowest-end processor being optimized for simplicity. ARM's unique ecosystem has enabled companies like Energy Micro to specialize in delivering energy-efficient processors targeted for the Internet of Things.

We have developed a testbed with supporting software that extends the Energy Micro Tiny Gecko starter kit [56] with acoustic sensing capabilities [57]. We denote this as the *EFM testbed*. Table 3.3 lists the average power consumption of the same functionalities that were implemented on the CheetahCub testbed. Here, voltage-frequency scaling was not employed. The

Table 3.3: Average Current Consumption on the EFM Testbed

	Simple Model (μA)	**Accurate Model (μA)**
Microphone	200	200
Pre-amp	260	260
ADC	510	510
LRT	160	2400
Total	**1130**	**3370**

supply voltage was fixed at 3.3V and the processor was configured to run at 32MHz.

Without voltage scaling, the simple processing on the CheetahCub testbed would consume 850μA; thus comparing the two testbeds, we see that the MSP430 is still more efficient for simple processing tasks. On the other hand, for computationally intensive processing, the ARM Cortex-M3 is 2.5$\times$ more efficient than the TI MSP430.

The Energy Micro processors have some promising features for "high performance" computing on microcontrollers:

1. 32-bit instructions along with ARM C compilers enables efficient code optimizations that can actually lead to more compact code than the TI MSP430 [58].

2. High performance is achieved using fixed-point processing with a dedicated 32x32 multiply instruction. In comparison, the high-end TI MSP430 utilized in the CheetahCub testbed has a *memory-mapped* 32x32 hardware multiplier unit.

3. ARM's unique ecosystem enables applications to leverage a growing code base and existing libraries (e.g., the CMSIS DSP library which works with any ARM processor).

4. Sleep power consumption is becoming competitive compared to the TI MSP430.

5. Specialized low-energy peripherals are designed for sensing applications.

On this last point, the low-energy peripherals enable application developers to mix hardware and software to achieve significant reduction in power consumption that could not be easily realized otherwise. To this end, the EFM testbed supports an efficient approximate computation of an energy detector made possible by using the on-chip analog comparator [57].

3.5 A Nano-power Acoustic Sensor

As demonstrated in both Table 3.2 and Table 3.3, the hardware components involved in acoustic data acquisition consume a significant amount of energy, especially when considering applications that require always-on listening. Specifically, for simple processing, data acquisition consumes 89% of the energy on the CheetahCub testbed and 86% of the energy on the EFM testbed!

This is a significant amount of energy, considering that in many applications, data analytics or high-level summary information may be more interesting than the raw acoustic data itself. The resource management framework presented in Chapter 2 affords us the ability to relax the design specifications for data acquisition. The optimization problem explicitly accounts for the increased uncertainty introduced by low-quality sensors, and generating optimal scheduling policies for specific applications demonstrates whether or not a newly proposed sensor design leads to more efficient systems.

Relaxing design specifications allows for innovative design trade-offs. In this section, we present an acoustic sensor that trades in signal fidelity not only for significantly reduced energy consumption, but also for manufacturability, using only a handful of commercially available components.

3.5.1 Low-power sensor design

As data acquisition dominates power consumption, we propose to 1) replace the ADC with a comparator, 2) eliminate pre-amplification, and 3) use a commercially available hearing aid microphone that consumes significantly less power.

Intuitively, the proposed sensor is designed to threshold the acoustic waveform at a value above the noise floor, which effectively triggers only when a high amplitude waveform is present. Although this strategy implies that we would achieve better performance by thresholding the *envelope* of the acoustic waveform, our desire to eliminate amplification makes rectification difficult to accomplish. Thus, as shown in Fig. 3.7, the proposed sensor performs a one-sided test directly on the acoustic waveform.

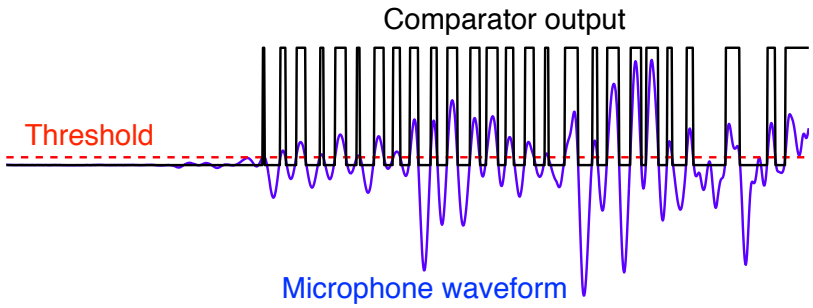


Figure 3.7: Illustration of the functionality of the proposed sensor.

In the actual circuit, which is shown in Fig. 3.8, the comparator is triggered when the acoustic signal is *below* the threshold, with the threshold set to be

below the noise floor. Because the data are assumed to be symmetric about

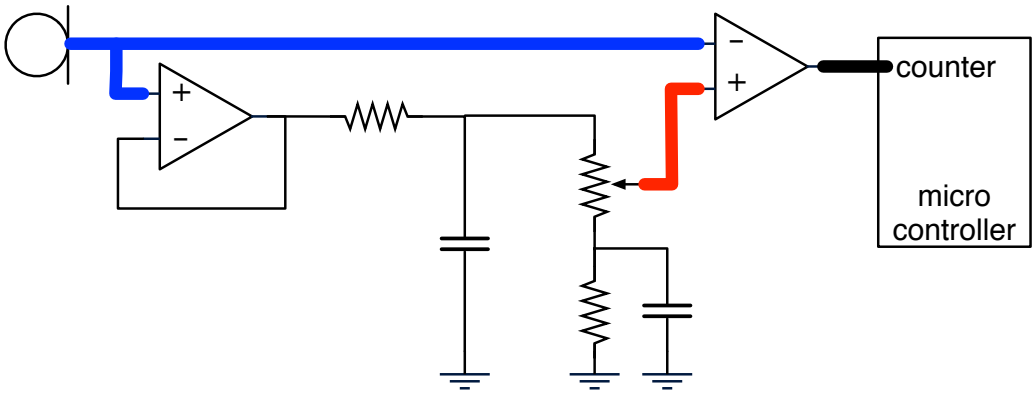


Figure 3.8: Schematic of the proposed acoustic sensor.

the mean, this one-sided test achieves the same performance as the one-sided test shown in Fig. 3.7.

The circuit exploits the fact that a MEMs microphone output has a built-in DC bias of around 0.7V. Instead of eliminating the DC voltage with a blocking capacitor, as is typically done in a microphone circuit, we buffer and low-pass filter the microphone waveform itself, which preserves the bias voltage. This bias is then used as the input to the voltage divider that is used to supply the comparator threshold. The top resistor in the voltage divider is a digital potentiometer, controlled in software to track and stay below the envelope of the noise floor.

Because the microphone waveform is not amplified, the dynamic range of the threshold must be tuned carefully. The largest value that the threshold can have is derived by tracking the mean of the microphone output, and the smallest threshold value is determined by the bottom resistor in the voltage divider; for instance, with the configuration in Fig. 3.8, the smallest value is about 16 mV offset from the mean[2].

[2]This value also takes into account the input impedance of the comparator, which was measured to be approximately 10 MΩ.

The comparator output should not be used to trigger a microcontroller to wake up and count, as this counting procedure would dominate the power consumption of the sensor. Instead, we borrow a clever counting design proposed in [55], and use the comparator output as an external clock for a counter, which is a peripheral commonly found on microcontrollers. In this way, the processor is not constantly being woken up, and given a maximum counting rate of 4kHz, the power consumption of the counter is projected to be at most 35nA, a negligible amount compared to the sleep current of the microcontroller, which is orders of magnitude higher.

Every 500ms, the microcontroller is woken up to 1) record the accumulated count value, and 2) reset the counter for the next time frame. These infrequent periodic "housekeeping" tasks consume very little overhead, as the consumed energy is amortized over the entire period. Thinking ahead, the inference task and sensor scheduling can be considered as additional housekeeping tasks, and for small state spaces, also contribute very little overhead.

3.5.2 Power profiling

Table 3.4 shows the power consumed by the different components of the proposed sensor.

Table 3.4: Power Consumption of Proposed Acoustic Sensor

Component	Avg. Current (μA)
ADMP401 mic [*ADMP801*]	200 [*17*]
AD5602 digipot	14
TLV2702 opamp + comparator	1.5
bias current	0.15
EM μC counter	0.035
Total	215.685 [32.685]

In this new setup, the microphone element clearly dominates power con-

sumption. Advances in MEMs microphone technology, such as the new ADMP801 [59], enable a 10x reduction in power. Although nano-power digipots such as the MAX5161 do exist, issues with mega-ohm resistors in the divider network [60] resulted in the current design, which uses the higher-power digipot.

A photograph of the prototype is shown in Fig. 3.9. The lower half of the board contains the proposed circuit, and the upper half contains a standard microphone preamplifier circuit. The MEMs microphone is connected via a daughterboard and is not shown.

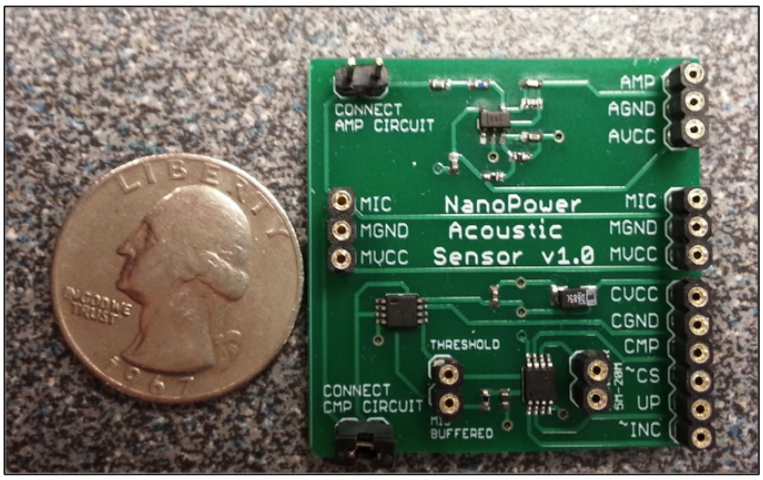


Figure 3.9: PCB of the proposed acoustic sensor.

3.5.3 Non-ideal noise sources

Aside from the one-bit quantization, we have observed that another source of performance degradation is the noisy value of the actual comparator threshold. We expect that this is due to 1) the sensitivity to noise due to the microphone signal being in the mV regime, and 2) the threshold being derived from the signal itself, implying that fluctuations occur because the first-order analog low-pass filter (which tracks the mean) is nowhere near ideal.

CHAPTER 4

SYSTEM DESIGN

We are now in a position to specify all of the components that are required for system design. We begin by characterizing and modeling the application dynamics and goals; we then construct observation models for the sensing actions built and proposed in Chapter 3, and generate optimal resource scheduling and inference policies to carry out the application task.

4.1 Temporal Dynamics

4.1.1 A hierarchical Markov model

To learn the temporal dynamics of the application, we utilize domain knowledge to create initial estimates of the transition matrix $\tau(x, x')$. Given the assumptions of the acoustic model in Section 3.2, we use three states to model the duration of the type A acoustic call, as shown in Fig. 4.1. Three states

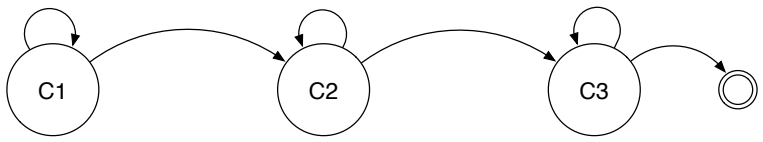


Figure 4.1: Markov chain used to model the beginning, middle, and end of a call. Transition structure shown here was learned, during which silence states were included before and after the call states; these silence states are not shown here.

are used because the call lasts about 1.5 seconds. The transitions shown in

Fig. 4.1 are based on the model structure that was learned from training data. Furthermore, the observation models for each calling state are not tied because acoustic frames typically contain less signal power at the beginning and end of a call, due to the silence that can be included in the beginning and end frames.

The arrival of acoustic calls shown in the previous chapter in Fig. 3.2 is modeled using a two-level hierarchical Markov chain. The top level, shown in Fig. 4.2, distinguishes between when the bird is present and when it is absent.

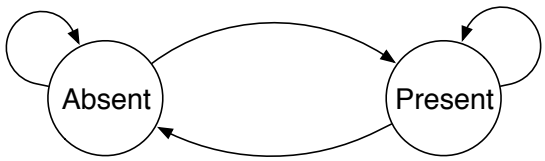


Figure 4.2: A simple two-state Markov chain to model when a bird is in the vicinity of the sensing device.

The duration times are characterized with parameters α and β, corresponding to the average durations of inactivity and activity, respectively; these values will be very close to 1 since we assume a time-step of 500ms and a bird typically remains in the vicinity for 15-45 minutes at a time. Due to the lack of sufficient training data, α and β, which quantify the "rare-event" assumption, are not learned. Instead, these parameters can be varied to study the effects of rare-event assumptions that are usually implicitly made in applications where wakeup mechanisms are utilized.

The second level only pertains to when a bird is present, and models the time between calls. Fig. 4.3 shows the empirical and fitted distribution of duration times between calls. With an average waiting time of 8.5 seconds and a call duration of 1.5 seconds, the average rate of calling is 10 seconds.

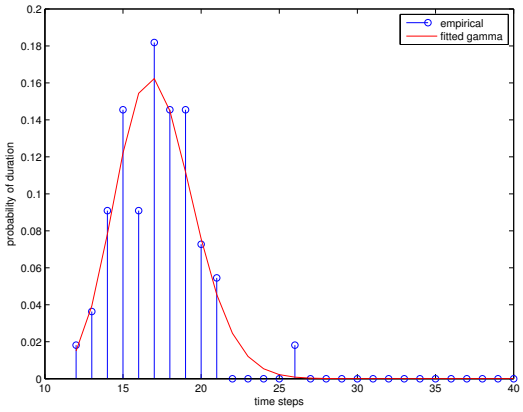


Figure 4.3: Empirical and fitted gamma distribution of duration times between calls when a bird is present (i.e., actively calling). Average waiting time before the next call is about 8.5 seconds, with the call lasting for about 1.5 seconds.

As this distribution is clearly not exponential, we initialize[1] to the model with the variable-duration model [61] shown in Fig. 4.4. The transition

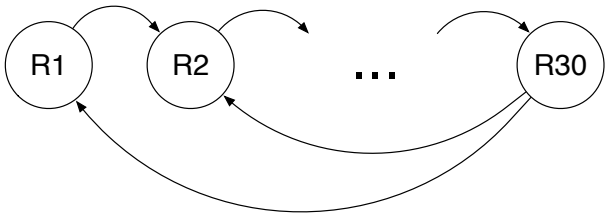


Figure 4.4: A variable-duration model of the times between calls.

probabilities from state R30 have the pmf determined by the fitted gamma distribution shown in Fig. 4.3.

The full hierarchical Markov model is shown in Fig. 4.5. Although inference can be performed directly on the hierarchical model [8], we flatten it

[1]In practice, a small non-zero probability is added to the other transitions. The learned transition matrix was manually checked to ensure convergence is as expected.

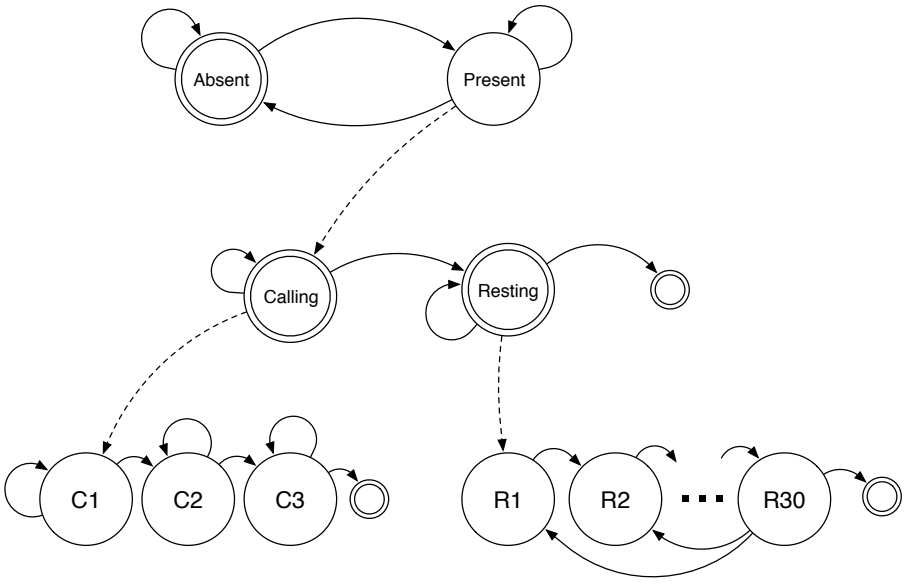


Figure 4.5: The fully augmented hierarchical Markov model.

to a standard Markov model because the lack of shared sub-models implies that there is little computational efficiency to be gained by preserving the hierarchical structure. Furthermore, flattening the model to a standard HMM enables us to use existing POMDP solvers[2].

In our application, the main benefit of the hierarchical interpretation is the abstract grouping of states. In Fig. 4.5, double-ringed states are *production* states, capable of directly generating observations. Single-ringed states are *abstract* states that generate a sequence of observations by activating the sub-models, denoted by the dotted lines. This abstraction will be useful in Section 4.2 when defining application goals and decision spaces.

[2]Developing POMDP solvers that can exploit hierarchical structure and reuse sub-models is left for future work involving applications with richer more complex structural connections.

4.1.2 Other modeling options

A hierarchical HMM is not the only extension of an HMM. Other options include the variable-duration HMM [61], semi-Markov model [62], stochastic context-free grammars, and in general, dynamic Bayesian networks [63]. Throughout this thesis, we have assumed discrete time steps, which may be natural for applications that extract periodic features based on frames, but is a limitation since most physical phenomena arrive in continuous time, and our assumption does not allow for variable frame sizes.

4.2 Application Goals

Within our sequential decision-making framework, application goals are expressed by defining an appropriate decision space $\mathcal{D}$, and then specifying a reward function to encourage certain behavior.

Consider the following three different application goals:

1. *monitoring for bird presence*: is a bird present or absent?
2. *monitoring for bird activity*: if it is present, what is it currently doing?
3. *identifying individual bird calls*: to count the number of calls heard in one day, for example.

In this chapter, we construct the POMDP problems for all three applications, and in Chapter 5 we analyze the performance trade-offs in monitoring for bird presence.

4.2.1 Monitoring for bird presence

In this scenario, one might simply be interested in knowing if a bird is currently within the vicinity of the sensing device. In this case, we define the decision space to be $\mathcal{D} = \{P, A\}$, which correspond to the abstract Present state and Absent state in Fig. 4.5. An equal error rate criterion would result in the following reward function:

$$R(x, d) = \begin{cases} 1 & \text{if } x \in X_d, \\ 0 & \text{else} \end{cases} \tag{4.1}$$

where

$$X_P = \{x \in \mathcal{X} : x \text{ is a sub-state of Present}\} \tag{4.2}$$

$$X_A = \text{Absent} \tag{4.3}$$

That is, for a decision d, X_d is the set of children states belonging to state d.

4.2.2 Monitoring for bird activity

In applications where we want to track the bird's current activity, we not only have to monitor for the presence of a bird, but also distinguish between when the bird is calling and when it is resting. To achieve this, we set the decision space to be $\mathcal{D} = \{C, R, A\}$, which consists of the abstract states in the hierarchical Markov model, along with the Absent state. An equal error rate criterion would result in the following reward function:

$$R(x, d) = \begin{cases} 1 & \text{if } x \in X_d, \\ 0 & \text{else} \end{cases} \tag{4.4}$$

where a reward is given when the current state x belongs to the decision's subset.

A special case of this application is the binary hypothesis-testing problem, where hypothesis H_1 corresponds to a bird calling, and the null hypothesis corresponds to no bird calling. For this application, the decision space could be set to $\mathcal{D} = \{H_1, H_0\}$, where $X_{H_1} = X_C$ and $X_{H_0} = X_R \cup X_A$.

4.2.3 Identifying bird calls

This application scenario demonstrates how our resource management framework can be used to generate policies that can make a single declaration for a single call spanning multiple time steps. This would be useful to count and timestamp each call, which could alternatively be done heuristically by post-processing (e.g., majority voting) the decisions from the previous application.

This final scenario is included because it illustrates that 1) decisions do not have to be a subset of the state space, and 2) the state space can be artificially augmented such that sensing actions influence state transitions, which is a feature of POMDPs, but not included in our original problem formulation. Our use of standard POMDP solvers makes this problem straightforward to solve.

Let $\mathcal{D} = \{defer, C, reset\}$, where C is declared only once per call (e.g., at the end of the call), and a decision *deferred* at all other times. Once a call is declared, the system must choose to *reset* before C becomes a valid decision option again, which ensures that only a single declaration is made per call. To support this mechanism, we introduce a deterministic state machine $F = \{0, 1\}$, where $F = 1$ when C is decided; this transition is enforced by

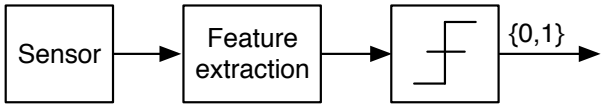


Figure 4.6: Real-valued features are thresholded, resulting in binary features.

properly defining the transition function when action $a = (C, s)$ is taken, for any sensing action s. When $F = 1$, a negative reward is assigned to deciding C to prevent the optimal policy from choosing this decision. A *reset* will transition the state to $F = 0$, and enable the system to declare another bird call.

4.3 Sensing Actions and Energy Consumption

Chapter 3 presented the development of four sensing actions, where each sensing action is defined by a unique configuration of sensing hardware, acquisition peripherals, and feature extraction algorithms for detecting GCW acoustic calls. The features computed by each sensing action are generally real-valued statistics. We consider sleeping as a sensing action, where the observation that is generated is an erasure, and provides no information about the state of the world.

In this thesis, we quantize the features into binary classes, as shown in Fig. 4.6. Thresholding is feasible because the test statistics that are generated are one-dimensional features (i.e., energy or count value). Because all sensing actions result in a binary "observation", the observation space is defined to be $\mathcal{Y} = \{0, 1, \epsilon\}$, where ϵ denotes an erasure, only used by the sleeping sensing action. The implications for quantizing features are discussed in the next section on observation modeling.

Table 4.1 provides a summary of each sensing action and the average cur-

rent consumed by each action.

Table 4.1: A Summary of the Sensing Actions Implemented on Two Testbeds

Sensing Action	Hardware	Software	Average Current (μA)
Sleep CC	none	none	3
Count CC	NPAS, counter on CheetahCub	thresholding	36
Energy CC	HD Mic, ADC on CheetahCub	energy + thresholding	720
Bandpass CC	HD Mic, ADC on CheetahCub	filter + energy + thresholding	8400
Sleep EFM	none	none	10
Count EFM	NPAS, counter on EFM	thresholding	43
Energy EFM	HD Mic, ADC on EFM	energy + thresholding	1130
Bandpass EFM	HD Mic, ADC on EFM	filter + energy + thresholding	3370

While our problem formulation allows for state-dependent sensing costs as described in Section 2.2.5, empirical experiments show that the average current draw is not affected by the acoustic activity level. Thus, the numbers in Table 4.1 fully characterize the sensing-action usage costs.

4.4 Observation Models

For finite action spaces, optimal decision policies that operate on continuous observations (i.e., real-valued features) actually result in belief-dependent quantization of the observation space [29]. Thus, the thresholding operation in Fig. 4.6 can be viewed as suboptimal because the thresholds and number of quantization levels are not belief-dependent. The disadvantage in solving for policies with continuous observations is that state-of-the-art policy solvers are slow, requiring stochastic simulation or Monte Carlo integration to evaluate

the value function within each iteration.

The disadvantage in suboptimal feature quantization is that a heuristic strategy for setting thresholds is required. The strategy we adopt is to maintain a constant "false alarm" rate. That is, the thresholding strategy was motivated by binary detection of whether or not a bird is calling (i.e., is the acoustic environment quiet or loud?). Thus, the threshold is set by allowing a fixed percentage of resting and absent periods to be falsely labeled as loud.

With discrete observation space $\mathcal{Y}$, the observation models consist of computing $P_s(y|x)$, where $y \in \{0, 1\}$, $x \in \mathcal{X}$ is the state space, and $s \in \mathcal{S}$ is the sensing action. For $y = 1$ and $x \in \{X_A \cup X_R\}$, this probability can be interpreted as the false alarm rate of "detector" s:

$$P_s(y = 1 \mid x \in \{X_R \cup X_A\}) = \Pr(T_s > \tau_s | \text{ bird is resting or absent}) \quad (4.5)$$

where T_s is the feature computed by sensing action s, and τ_s is the threshold set to achieve some constant false-alarm rate. As discussed in Section 3.2, these probabilities reflect the quality of each sensing action, and fundamentally depend on the acoustic model parameters, signal power K and noise power σ_n^2.

For the performance evaluation described in Chapter 5, instead of actually modeling the data as Gaussian, as was done in Section 3.2, we learn the observation probabilities (e.g., (4.5)) from clean unlabeled data according to the following procedure:

input : Noise power σ_n^2, and clean unlabeled data.
output: Transition matrix $\mathcal{T}$, and observation matrices $\{\mathcal{O}_s^{\sigma_n} : s \in \mathcal{S}\}$.

1. Find transition matrix $\mathcal{T}$ and observation matrix $\mathcal{O}$. This is done by 1) computing the band-pass energy feature on the clean data, and 2) running the Baum-Welch algorithm to learn $\mathcal{T}$ and $\mathcal{O}$.
2. Label the training data. This is done by running the Viterbi algorithm using the generated features, $\mathcal{T}$ and $\mathcal{O}$ from step 1.
3. Add white Gaussian noise with variance σ_n^2 to the clean training data. In this thesis, we do not consider other types of noise (e.g., wind noise, airplanes, interference from other animals).
4. For each sensing action, generate the noisy observations. This consists of computing the features and setting the threshold τ_s such that 10% of the Absent and Resting frames are declared as "loud".
5. Use the labels to compute the ML estimate of the observation matrices, $\{\mathcal{O}_s^{\sigma_n} : s \in \mathcal{S}\}$.

Algorithm 1: Learning transition and observation matrices from clean unlabeled data with added noise.

4.5 Run-time Operation

Sections 4.1 – 4.4 explicitly defined the POMDP problem $\mathcal{M} = \langle \mathcal{X}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Y}, \mathcal{O}, \gamma \rangle$, where $\mathcal{A} = \mathcal{D} \times \mathcal{S}$. The choice of discount factor γ is discussed in Section 5.2. See Table 4.2 for specific details.

The POMDP problem $\mathcal{M}$ generates optimal policy ϵ, which is found using the Heuristic Search Value Iteration (HSVI) solver [27], freely available for download at [64]. The policy ϵ is characterized by matrix V_ϵ whose columns are the α-vectors representing the optimal value function such that

$$V^*(\mathbf{b}) = \max \; V_\epsilon^\top \mathbf{b} \tag{4.6}$$

and a vector A_ϵ where each element is an action corresponding to the α-vector in V_ϵ. Thus, the optimal action to take for belief $\mathbf{b}$ is the action in

Table 4.2: POMDP Definition for the Decision Process Involving Inference and Sensing Actions for Acoustic Wildlife Monitoring

Symbol	Definition	Defined in ...
$\mathcal{X}$	state space	Section 4.1
$\mathcal{T}$	transition matrix	Section 4.1, learned in Algorithm 1
$\mathcal{D}$	decision space	Section 4.2
$\mathcal{R}$	reward function	Section 4.2
$\mathcal{S}$	sensing actions	Chapter 3, summarized in Section 4.3
$\mathcal{C}$	sensing energy costs	Chapter 3, summarized in Section 4.3
$\mathcal{Y}$	observation space	Section 4.4
$\mathcal{O}$	observation matrices	Section 4.4, learned in Algorithm 1

the index of A_ϵ corresponding to the α-vector that achieves the maximum in (4.6). Since there is a one-to-one mapping from $\mathcal{A}$ to $\mathcal{D} \times \mathcal{S}$, the optimal inference decision and sensing action can be recovered.

At run-time, the system operates as follows. It is assumed that a prior $\mathbf{b}_0$ and sensing action s_1 have been specified. Time starts at index $t = 1$:

1. Generate an observation y_t using the sensing action s_t.
2. Incorporate observation to update belief $\mathbf{b}_{t-1} \rightarrow \mathbf{b}_t$.
3. Generate action $a_t = (d_t, s_{t+1})$ by executing (4.6).
4. Repeat from step 1 at next time step.

CHAPTER 5

PERFORMANCE EVALUATION

5.1 Evaluation Procedure

In this chapter, we focus on comparing different scheduling policies. This is accomplished by maintaining the belief state and using the optimal inference decisions, even for the heuristic scheduling architectures. The only difference is that the heuristics do not utilize the belief state to make scheduling decisions.

Performance evaluation is done by stochastic simulation. Fig. 5.1 shows the steps for generating sample paths and observations while running the sequential decision process.

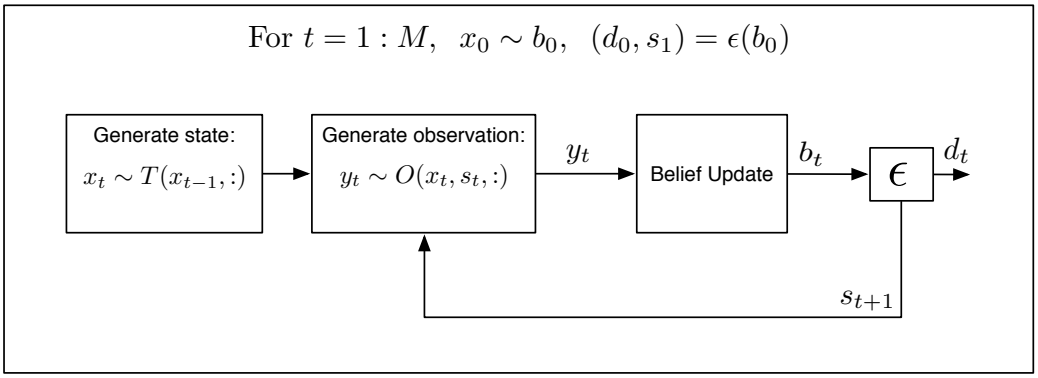


Figure 5.1: Evaluating a scheduling policy ϵ consists of executing the block diagram for M time steps. The state x_t is hidden from the system, but is used to generate observations and to evaluate the reward received by deciding d_t.

By keeping track of the true states, decisions, and sensors utilized, we

can compute the relevant time-average statistics, which converge to the true expectations due to the ergodicity assumption. The statistics that we are interested in are:

$$\hat{r}_M^{\epsilon} = \frac{1}{M} \sum_{t=1}^{M} R(x_t, d_t) \tag{5.1}$$

$$\hat{c}_M^{\epsilon} = \frac{1}{M} \sum_{t=1}^{M} C(s_t) \tag{5.2}$$

Sweeping the Lagrange multiplier in (2.58) generates a sequence of policies, $\{\epsilon_\lambda\}$. Evaluating each ϵ_λ results in a different performance-reward/energy operating point; together, the operating points generate the reward/energy trade-off curve that we use in our comparisons.

5.2 Setting the Discount Factor

Although we optimize a discounted criterion using a Lagrangian relaxation, we report the performance as a time-average reward/energy trade-off, since this is a more natural criterion for energy consumption. Although there is no guarantee that the reward/energy trade-off we label as "optimal" is actually optimal for an average-reward criterion with an average-energy constraint, we have observed that a large enough γ appears to capture the long-term average value of each action. For the experiments in this chapter, we set $\gamma = 0.9999$.

5.3 Synergistic Design

There is a synergy created by applying control theory to make the design of systems in embedded signal processing applications principled. In particu-

lar, the theory abstracts away the otherwise complex process of determining how to best utilize developed resources and enables us to relax design specifications, make design trade-offs we otherwise would not, and quantify the utility of new designs with respect to application goals.

As quantified in Fig. 5.2, our application of this methodology to the proposed low-power acoustic sensor has the potential to reduce energy consumption by an order of magnitude, relative to the original application running on the CheetahCub testbed, which was already designed for energy efficiency. The original software implemented a cascade of signal models for energy-

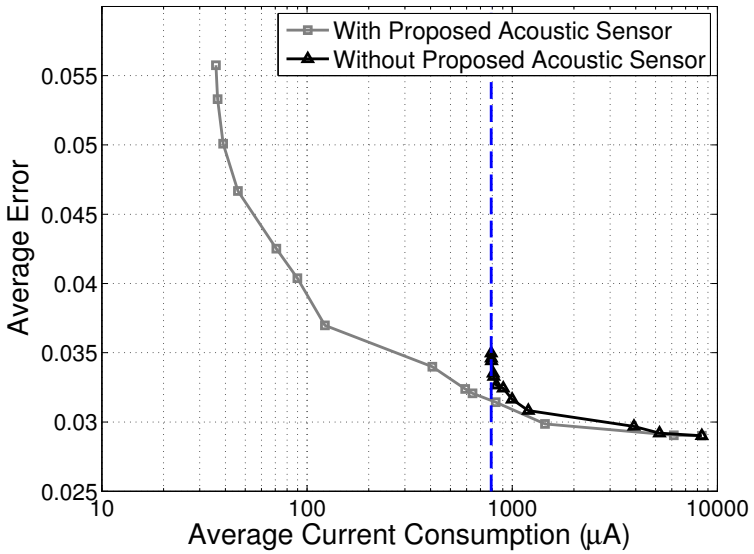


Figure 5.2: Demonstrating the synergy from cross-discipline design. Our control-theoretic framework enables significant energy savings by optimally utilizing a novel acoustic sensor that achieves microwatt power consumption by relaxing design specifications.

aware detection of the GCW bird calls [65]. The cascade operates by running the simple energy detector until something interesting occurs, at which point the block of data is re-processed with more sophisticated processing. Appendix A shows how to control the detector thresholds to make the cascade energy-aware. Unfortunately, the cascade architecture, which is always

listening with the high-fidelity acoustic sensor, hits a bottleneck due to data acquisition, as illustrated by the blue dotted line in Fig. 5.2. As the figure illustrates, when energy is at a premium, designing a new sensing action can unlock energy savings that the utilization theory alone would not reveal.

5.4 Comparing Different Testbeds

In Chapter 3, we presented the EFM testbed, which was shown to be more energy efficient than the CheetahCub testbed when high processing performance is required. Profiling the power consumption of different sensing and processing tasks demonstrated that neither testbed was always more energy efficient. In particular, the CheetahCub testbed, built around the TI MSP430, is better suited for simple processing and is slightly better at sleeping. Fig. 5.3 quantifies this trade-off within the context of the inference application.

5.5 The Efficiency of the Wakeup Mechanism

We study the efficiency of intuitive design. The stochastic control framework allows us to quantify how much energy savings are being "left on the table" by not making optimal use of the resources and information available to the system. The wakeup mechanism we employ utilizes the low-power acoustic sensor as its always-on sensor. When the *Count* sensing action declares $y = 1$, the *Bandpass* sensing action is triggered to run for the subsequent two frames, after which the low-power acoustic sensor takes over again. Two frames were chosen because we know that a bird call lasts for 3 frames on average.

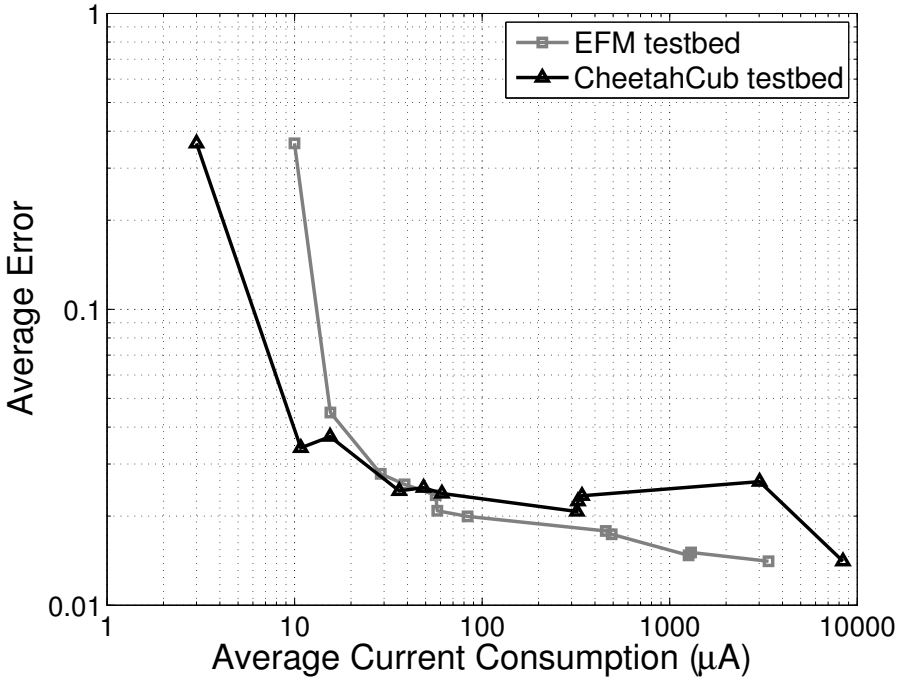
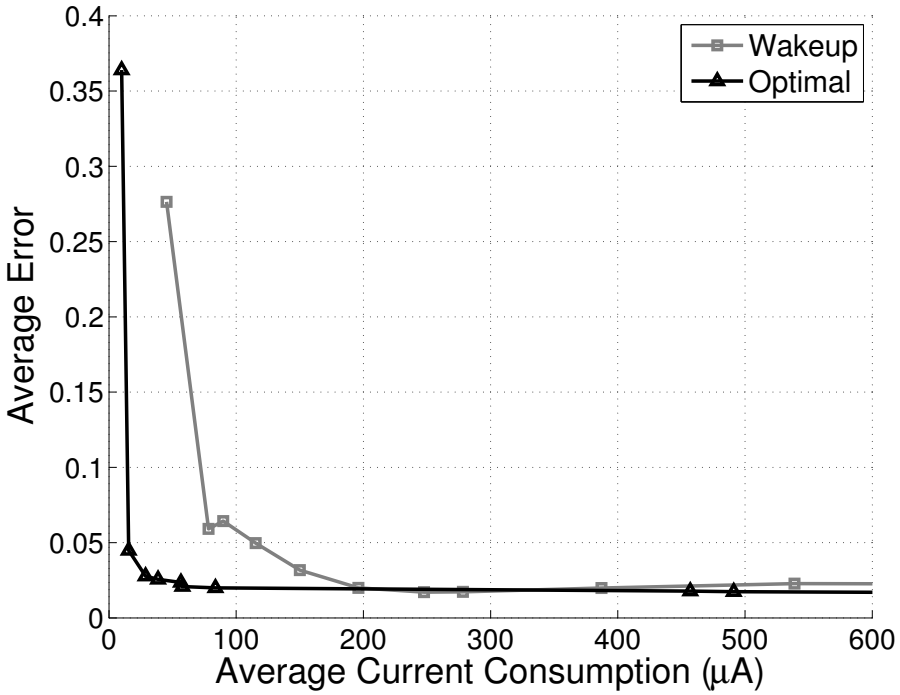
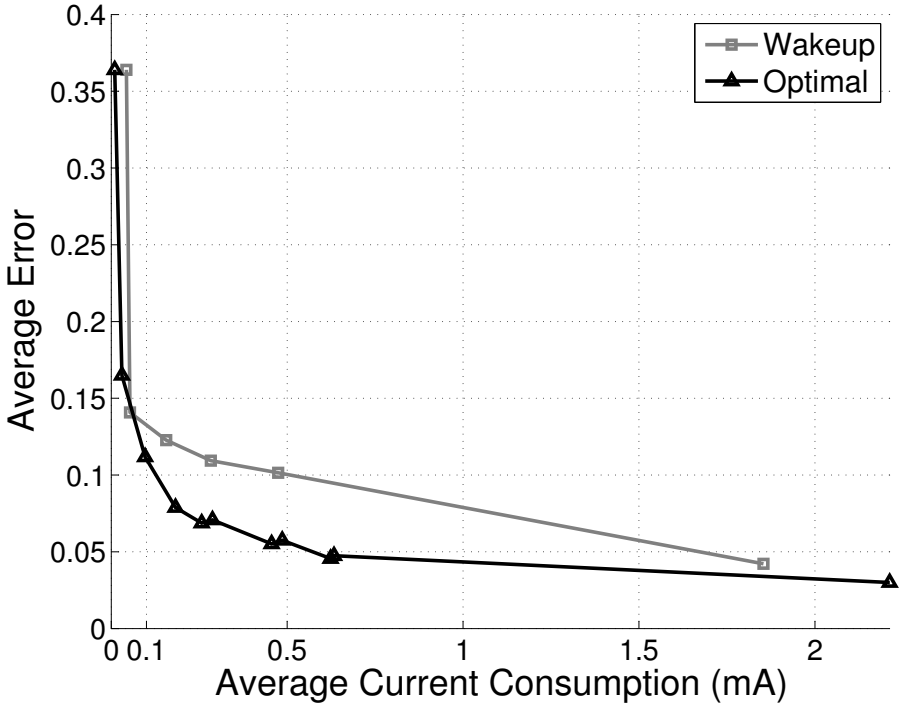


Figure 5.3: Application-specific trade-off between the EFM and CheetahCub testbeds.

Fig. 5.4 shows the comparison between the optimal policy and wakeup mechanism running on the EFM testbed for high SNR and low SNR scenarios. The application scenario presented here is monitoring for bird presence. When operating with average current consumption under 500μA, a significant gap in performance is seen. Furthermore, although not shown in the figure, even at high energy rates, the wakeup mechanism performs worse than optimal – this gap arises because the wakeup mechanism forces the device to reset to the low-power acoustic sensor, and may be an artifact of the particular heuristic we chose to implement.



(a) High SNR ($\sigma_n^2 = 0.02$).



(b) Low SNR ($\sigma_n^2 = 0.04$).

Figure 5.4: Performance/energy trade-off in monitoring for bird presence. In high SNR, the optimal policy is able to make efficient use of the low-power acoustic sensor to deliver high performance at stringent energy rates, relative to the wakeup mechanism. In low SNR, there is a significant gap at moderate energy rates (from 100μA–1mA).

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

This thesis challenges conventional wisdom which is that when energy is limited, a device must be forced to sleep; adopting this strategy would fail to capture dynamic events from "information-rich" sensing modalities, such as image or acoustic sensors. This thesis has demonstrated one way that systems can deliver high sensing performance while consuming energy that is dramatically lower than what is expected today. The key insight is to match device resources to environmental dynamics, and this thesis provides a systematic principled approach to study the performance/energy trade-offs that can be achieved in such applications.

6.1 Conclusions

Run-time control of sensing resources in dynamic uncertain environments is a powerful idea rooted in stochastic control theory. In energy-limited systems today, sensing and processing architectures are heuristically designed to be energy efficient, which is achieved by manually exploiting problem characteristics.

Applying fundamental stochastic control results to manage sensing resources on energy-limited systems in dynamic applications leads to a principled approach; we demonstrate that when energy is scarce, heuristic designs do not necessarily make the best use of device energy, which is when it is

most critical to do so. In general, prior to our work there was no way to numerically quantify the performance gap arising from heuristic utilization of sensing resources.

Part of this inefficiency in using heuristics arises because fixed sensing architectures create a "division of labor" enabling researchers to simplify a complex problem into manageable parts and work on improving a single aspect of the problem. Unfortunately, this creates artificial boundaries preventing innovative aggressive design trade-offs that could potentially lead to huge energy savings. As a case in point, to enable always-on acoustic inference applications in the ultra low-power regime, our work points to the biggest impact coming from innovations in sensing hardware, where the design trade-offs should be driven by application-level goals instead of intermediate activity-detection performance.

While it is well known that in order to achieve high energy efficiency, system-level goals should permeate through all layers of design, what is less obvious is how to proceed with system design in dynamic inference applications. This thesis lays the foundation and demonstrates the process for one way to accomplish this task. In particular, the abstraction provided by the control theoretic framework easily cuts through many of the artificial boundaries that researchers have constructed for themselves. Using the abstract notion of a sensing action, our proposed acoustic sensor is a simple hardware/software solution that demonstrates that one can innovate by making more aggressive design trade-offs, and then systematically quantifying the utility of the relaxed design with respect to application goals. This is particularly powerful for applications exhibiting dynamics, stochasticity, and uncertainty, all of which are hard to quantify and analyze if designing heuristically, but precisely handled using stochastic control theory.

Although the optimal policies that are generated demonstrate the achievable performance/energy trade-off for a given set of system resources, one should not be quick to assume that this optimal policy is the *best* policy to implement in practice. Two challenges for energy-efficient systems are that any practical scheduling architecture needs to be 1) extremely lightweight and 2) robust. Robustness is the more challenging issue, particularly because generative approaches are known to be brittle with respect to mismatches in model parameters. In that regard, simple intuitive heuristics are attractive since they work "well enough". The theoretical framework developed here can be used iteratively as a *tool* that systematically improves heuristic design by informing where the bottlenecks are in the system with respect to application goals.

Devices commercially available today such as the Energy Micro microcontrollers are very promising, as they exhibit the heterogeneous hardware capabilities that this thesis advocates. In addition to redefining design specifications, blurring the line between hardware and software will lead to innovative designs that can have significant impact on the energy efficiency of a sensing application.

The methodology presented in this thesis will become increasingly relevant as wearable computing and the Internet of Things become more prevalent. The constraints on form factor and size ensure that energy will continue to be a major challenge, especially in applications demanding high-quality information and context awareness.

6.2 Future Directions

A weakness of the performance evaluation presented in Chapter 5 is that the policies were not evaluated on real data traces. There were a couple of reasons for this, including a lack of sufficient testing data (over 400 hours of simulation data were used to generate the performance/energy plots), and the lack of adaptation in the algorithms. In particular, in the POMDP framework, the optimal policies are generated for a particular set of model parameters.

Bayes-adaptive POMDP [66] looks to be a promising theory for extending our framework to design optimal policies that incorporate the learning of unknown model parameters. The basic idea behind Bayes-adaptive POMDP is that these unknown parameters are hidden, and observations can be used to estimate them online along with the hidden states. Thus, selecting observations to explore vs. exploit becomes a part of the control policy, and augmenting the belief space with the distributions of the hidden parameters results in yet another, larger, POMDP.

We view the Bayes-adaptive-POMDP extension as being analogous to what POMDP was for heuristic wakeup design. In particular, constructing a policy that optimally explores vs. exploits will demonstrate the optimal performance that can be expected for systems that heuristically track parameters, but it can be expected that heuristic tracking may be preferred due to its simplicity and effectiveness. Another potentially interesting extension would be to combine our generative approach with discriminative approaches [42] that use training data to learn control policy mappings.

We have begun to apply the techniques and process introduced in this thesis to more challenging problems such as always-on speech recognition.

An obvious limitation of the proposed framework is that it cannot handle the Viterbi decoder, which is a common and useful technique for smoothing and correcting incorrect decisions based on patterns seen in the future. For these problems, while it is possible to show that the dynamic programming principle holds, the challenge is to find a computationally efficient algorithm to handle the resulting mixed discrete-continuous state space.

The application and scope presented in this thesis were somewhat limited as the quality of sensing actions were, for the most part, ordered with respect to energy costs. We predict that significant impact will be achieved in applications with multiple innovative sensing modalities and features specialized for particular scenarios or conditions. This would enable performance improvements and *robustness*, a potentially more compelling justification for the need for control.

Heterogeneous computing platforms with processors and accelerators specialized for specific functionalities are emerging as a popular technique to extend battery life (e.g., ARM big.LITTLE [67]) and increase performance. Identifying a critical application and applying our methodology to improve the efficiency of heterogeneous computing platforms could potentially have major immediate impact on today's mobile devices.

Finally, this thesis did not consider networked devices, where the value obtained from energy-constrained sensor nodes is ultimately derived not from a single device, but from the information that the aggregate collects. Our view up to this point has been that in order for interesting sensor networks to really take off, individual sensor nodes must be able to operate efficiently in information-rich sensing environments, which generally have context information that can be exploited to save device energy. Exploring how to schedule sensing resources on a single device given the goals and additional

knowledge obtained by networked applications will undoubtedly lead to exciting research questions.

APPENDIX A

CASCADING SIGNAL MODELS FOR ENERGY-AWARE DETECTION

In this appendix, we present an alternative formulation to the problem of monitoring the GCW bird calls. In particular, we consider the application of detecting the bird calls embedded in noise. This formulation does not explicitly exploit problem characteristics, such as the repetitive calling rate or probabilities of bird arrivals and departures.

The problem is considered as a binary detection problem, and the focus is on delivering the performance of sophisticated signal models at the energy consumption of their simpler counterparts. The basic strategy to enable this is to utilize a cascade architecture, which is useful for detecting the presence of a signal of interest. In the cascade architecture, the decision of an always-on low-complexity detector is used to trigger a more accurate, albeit computationally expensive, detector.

Although in principle it is clear that a cascade of detectors can be energy-efficient, in practice, the actual energy-efficiency depends on the operating point of the cascade. In particular, in the context of time-varying environments, unknown parameters, and potentially time-varying system constraints, a principled approach to operate the cascade that provides the best detection performance for a given energy budget is desirable. Algorithms with the ability to make this run-time energy/performance trade-off are known as "energy-aware" [68]. Any energy-aware algorithm for controlling a cascade of signal models must itself be energy-efficient and computationally

simple in order to minimize overhead.

A.1 Problem Formulation

The detection of the GCW call can be formulated as a binary hypothesis-testing problem, where H_0 corresponds to a noise-only environment, and H_1 signifies that the bird is calling. Thus, we consider the binary hypothesis testing problem, which is to map noisy sensor observations, $y \in \mathcal{Y}$, to a binary decision, H_0 or H_1. A signal and noise model implies the specification of observation densities $f_i(y)$. The optimal decision mapping is in the form of a likelihood ratio test (LRT):

$$\frac{f_1(y)}{f_0(y)} \underset{H_0}{\overset{H_1}{\gtrless}} \tau \tag{A.1}$$

where the left-hand side is the *likelihood ratio*, and τ is a threshold such that if the likelihood ratio is greater than τ, H_1 is chosen and a detection is declared. (A.1) can be thought of as a mapping $\delta : \mathcal{Y} \to \{H_0, H_1\}$ that is parameterized by the threshold τ. We use the terms 'mapping', 'LRT', and 'detector' interchangeably.

A.1.1 Cascade Operation

A block diagram of the cascade architecture for detection is given in Fig. A.1. The detection process in the cascade operates as follows:

1. The sensor collects an observation, or a set of measurements (collectively called an observation).

2. The signal model from the first stage is used to analyze the sensor observation and make a decision.

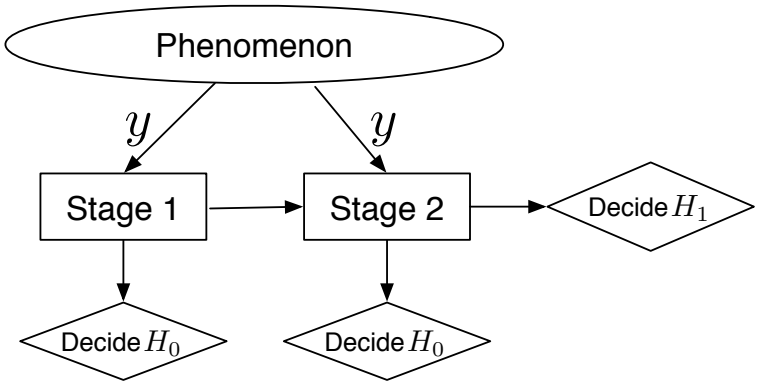


Figure A.1: Block diagram of cascade architecture for detection. A positive detection from an energy-efficient first stage triggers the use of a more accurate signal model in the second stage.

3. A positive detection from the first stage triggers the use of a more accurate signal model to analyze the *same* sensor observation and make a final decision.

4. If the first stage decides H_0, then the decision process terminates.

This scheme results in event-driven selection of the signal model to use for detection, and can be energy-efficient if the computational complexity in the first stage is low and the event of interest rare.

To be more precise, we make the following assumptions:

A1. Two signal models: 1) a simple but suboptimal model and 2) an accurate model.

A2. Detectors for each signal model are optimal likelihood ratio tests, denoted as mappings $\delta_1, \delta_2 : \mathcal{Y} \to \{H_0, H_1\}$.

A3. Energy cost to evaluate each LRT is fixed; denote these as c_1 and c_2, respectively.

A4. $c_2 > c_1$ (i.e. cost of evaluating δ_2 is greater than cost of evaluating δ_1).

A5. Two-stage cascade structure, which leads to the following system-level decision rule:

$$\delta(y) = \begin{cases} H_0 & \text{if } \delta_1(y) = H_0 \\ \delta_2(y) & \text{if } \delta_1(y) = H_1 \end{cases}$$

A6. The occurrence of H_1 is a rare event. More precisely, let $\pi_i \triangleq \Pr(H_i \text{ is true})$. Then, a rare-event assumption states that $\pi_1 \ll \pi_0$. This assumption ensures that the use of a cascade is well justified, since the first stage would be rendered redundant if the event of interest occurred with high frequency.

A7. No point-mass in the observation pdfs under any signal model. This is a technical condition that allows us to avoid considering randomized policies, for simplicity.

From these assumptions, we make the following observations:

O1. From (A2), detector thresholds τ_1 and τ_2 parameterize the mappings δ_1 and δ_2, respectively, and are sufficient to control detection performance.

O2. (A3) and (A5) imply that energy consumption of the cascade is a function of δ_1, and therefore, of τ_1 only.

Define Y_i to be detector i's individual *detection region*, which is the set of observations that get mapped to H_1 (i.e., $Y_i = \{y \in \mathcal{Y} : \delta_i(y) = H_1\}$). Then, the cascade's detection region is given as $Y_1 \cap Y_2$; the system-level detection performance of the cascade is:

$$P_d(\tau_1, \tau_2) = P(Y_1 \cap Y_2 \mid H_1 \text{ is true}) \tag{A.2}$$

$$P_f(\tau_1, \tau_2) = P(Y_1 \cap Y_2 \mid H_0 \text{ is true}) \tag{A.3}$$

where the dependence on thresholds is shown explicitly on the left-hand side.

The energy consumption of the cascade is determined by how often the first stage declares H_1 and is given by:

$$EC(\tau_1) = c_1 + c_2 \cdot P(Y_1) \tag{A.4}$$

$$= c_1 + c_2 \cdot (\pi_0 \cdot P(Y_1|H_0 \text{ true}) + \pi_1 \cdot P(Y_1|H_1 \text{ true})) \tag{A.5}$$

$$= c_1 + c_2 \cdot (\pi_0 \cdot P_f(\tau_1, 0) + \pi_1 \cdot P_d(\tau_1, 0)) \tag{A.6}$$

where $P_f(\tau_1, 0)$ and $P_d(\tau_1, 0)$ are the probabilities of false alarm and true detection for running only the *first* detector. To see why, note that $\tau_2 = 0$ implies that the second stage declares all observations as H_1 (i.e. $Y_2 = \mathcal{Y}$). Thus, $P(Y_1 \cap \mathcal{Y} \mid H_i \text{ true}) = P(Y_1 \mid H_i \text{ true})$.

A.1.2 Related Work

Wakeup mechanisms and decision trees as discussed in Chapter 2 are clearly relevant. [69, 70, 71] all optimize thresholds in a cascade, but fall under the category of distributed detection [72, 73] because observations are assumed to be independent. In this formulation, we assume that detector stages operate on the *same* sensor observation. Thus, it is more appropriate to consider our work as cascaded *signal models* for detection.

Incremental refinement [74] is a different strategy to develop energy-aware signal-processing algorithms. The idea is to incrementally scale the computational complexity of recursive algorithms and terminate early if the energy budget is exceeded. This idea has been applied in many areas of signal processing [75, 76, 77, 68].

A.1.3 Optimization Formulation

Given the assumptions and observations from the previous section, the problem is to find detector thresholds that optimize some criterion involving performance and energy. Motivated by applications in energy-limited devices with a potentially time-varying supply of energy, we consider the problem of maximizing detection performance, subject to an energy constraint. From the previous section, the detection performance is characterized by the true-detection and false-alarm rates of the cascade. Consider the energy-constrained Neyman-Pearson criterion [78]:

$$\begin{aligned} \max_{\tau_1,\tau_2} \quad & P_d(\tau_1,\tau_2) \\ \text{s.t.} \quad & P_f(\tau_1,\tau_2) \leq \alpha \\ & EC(\tau_1) \leq \beta \end{aligned} \tag{A.7}$$

for some $\alpha \in (0,1)$ and $\beta \in [c_1, c_1 + c_2]$. Constraining the false-alarm rate, as opposed to considering a weighted sum of the two types of decision error, is motivated by monitoring applications where the detector would have very little utility unless the false-alarm rate can be held under a fixed value, α.

A.2 Solution Characterization

There are two main results: 1) utilizing all of the energy is optimal, and 2) setting the false alarm rate as high as is allowed is optimal. The first statement holds because we assume that both stages operate on the *same* observation. The implication is that optimal thresholds can be found by sequentially satisfying the problem constraints.

For clarity in presentation, proofs for the theorem and supporting lemmas

can be found in Section A.4.

A.2.1 Utilizing All Available Energy

Theorem 2 *Let τ_1^* and τ_2^* be a solution to (A.7) with $P_f(\tau_1^*, \tau_2^*) \leq \alpha$ and $EC(\tau_1^*) < \beta$. Then, there exist a τ_1' and τ_2' such that:*

$$P_d(\tau_1^*, \tau_2^*) = P_d(\tau_1', \tau_2') \tag{A.8}$$

$$P_f(\tau_1', \tau_2') \leq \alpha \tag{A.9}$$

$$EC(\tau_1') = \beta \tag{A.10}$$

∎

This result states that without loss of generality, we only need to consider the solutions where all of the available energy is consumed.

Remarks:

1. If the second stage does not operate on the same observation as the first stage (e.g., in low-power wakeup mechanisms, the second stage operates on the *next* available observation), then the required analysis follows more closely to cascaded detectors in distributed detection [73], where a suitable optimization technique is required to find the optimal solution, which may not use all of the energy.

2. Even when environmental conditions change, Theorem 2 guarantees that using all the energy will still be optimal.

3. The theorem holds for general signal/noise models, as no assumptions about the actual distributions (besides continuity, (A7)) were made.

4. Theorem 2 will hold even if the second-stage signal model fails to actually be *uniformly* more accurate than the first (e.g. over all SNRs).

Theorem 2 results in the optimal δ_1^*. In turn, this fixes the detection region, Y_1^*, which is the set of observations that the second stage will get to see. Even with this restricted set of observations, the optimal test for the second stage is an LRT, with the threshold chosen to satisfy the false-alarm constraint with equality. This implies that τ_2^* is found by setting it such that the false-alarm rate is α.

A.2.2 Degeneracy Condition

A degeneracy occurs if $P(Y_1^*|H_0 \text{ true})$ is less than α. In this situation, the energy constraint was so stringent, relative to the false-alarm constraint, that it is optimal for the second detector to declare everything as H_1. Although not considered in this formulation, this degeneracy implies that if we expect extremely stringent or imbalanced system requirements, we should run a *generalized* cascade, where the system can decide to terminate the cascade early to make a final decision.

A.2.3 Problem Solution

Assuming the degeneracy is handled separately, finding optimal thresholds essentially boils down to solving two sequential one-dimensional root-finding problems:

$$EC(\tau_1) = \beta \tag{A.11}$$

$$P_f(\tau_1^*, \tau_2) = \alpha \tag{A.12}$$

Unfortunately, a closed-form expression for P_f does not exist, even with traditionally "simple" signal and noise models, due to shared observations between detectors. The problem is further exacerbated by the fact that signal-model parameters may be unknown and system constraints such as the available energy supply may be time-varying.

The seminal work by Robbins and Monro [79] uses a simple first-order successive-approximation technique to solve root-finding problems when the function is unknown, but "noisy" evaluations of the function are available; this general class of problems is known as *stochastic approximation* [80]. In our problem, the energy consumption and false alarm rate can be estimated by tracking the system's detection performance.

A.2.4 System Architecture

Our system has three periodic tasks: 1) detection, 2) performance-tracking, and 3) threshold-tracking. The detection and performance-tracking tasks have a period of 1 Hz, while the threshold-tracking task is executed once every 60 detection cycles, such that the thresholds are updated once a minute. The thresholds are tracked to satisfy (A.11) and (A.12). The software execution flow, which is triggered once per second, is shown in Fig. A.2.

A.2.5 Performance/energy trade-offs

There are several factors that limit the energy savings achieved by operating the cascade. Fundamental limits include the energy scalability of the system and the underlying activity level of the event of interest. For a given energy budget, SNR and false-alarm constraints determine the actual performance loss as compared to the accurate detector.

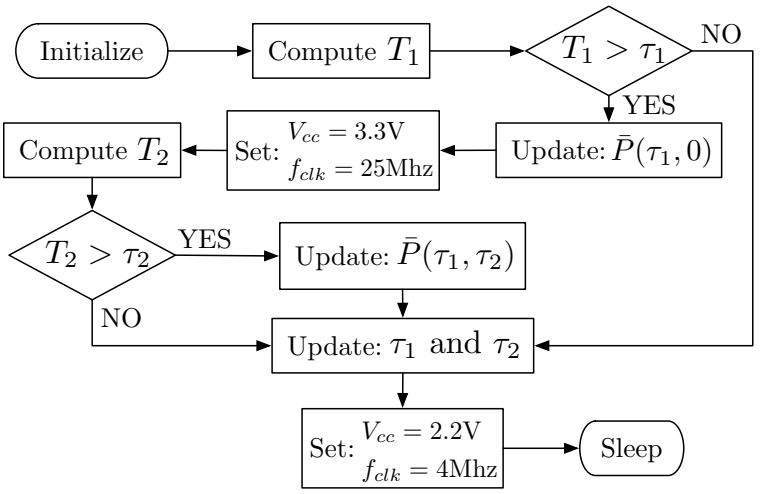


Figure A.2: Software execution after a block of data is collected. T_1 is the energy-detector statistic as computed in (3.1), and τ_1 is the associated threshold, tracked over time. T_2 is the BP-energy detector statistic as computed in (3.2) and τ_2 is its threshold. $\bar{P}(\tau_1, 0)$ is the average detection performance of the energy detector and $\bar{P}(\tau_1, \tau_2)$ is the detection performance of the cascade. These performance numbers are used to update τ_1 and τ_2.

To visualize the performance/energy trade-off, Fig. A.3 plots three ROC curves, where each curve represents the cascade performance at a different energy budget, β_i, for $i = 1, 2, 3$. The activity level was measured to be 2.6%. We see that if the false-alarm constraint is $\alpha = 10^{-3}$, then we can achieve energy savings of 16x, with no loss in true-detection performance. Furthermore, in the limiting case where the activity level goes to zero, the average power consumption of the cascade would be $c_1 + c_2 \cdot P_f(\tau_1^*, 0) = 1.356$ mW, and the energy savings would be 20x, compared to continuously running the BP-energy detector.

On the other hand, if the false-alarm constraint is 10^{-1}, then our energy savings would reduce to 3x, in order to achieve performance comparable to the BP-energy detector. In general, a system designer can use Fig. A.3 to understand and predict the performance/energy trade-offs associated with typical (β, α) constraints.

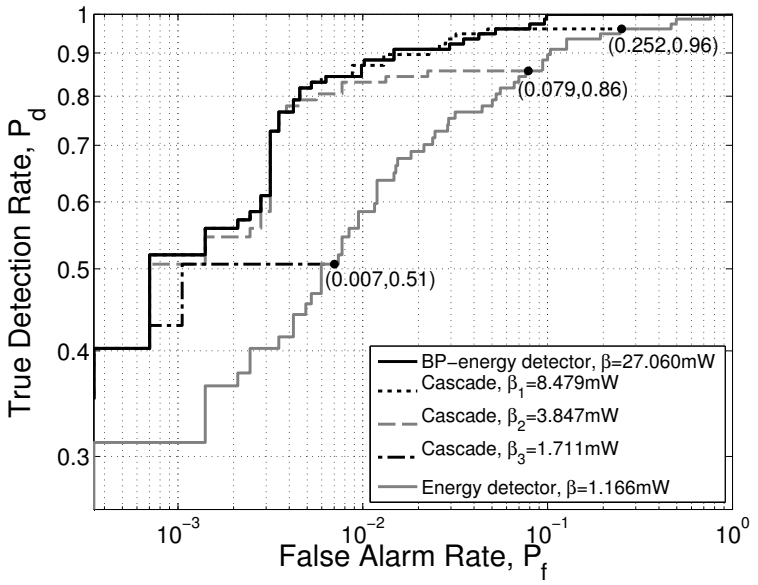


Figure A.3: The ROC curves of the energy-aware cascade for three different energy budgets, β_i. The individual ROC curves of the two stages have been re-plotted from Fig. 3.3; they are an upper and lower bound on the cascade performance. Note that the energy savings depends on the required false-alarm rate, P_f. The points $(P_f(\tau_1, 0), P_d(\tau_1, 0))$ are denoted for reference, and can be used to compute the energy budget of the associated curve, along with easily identifying degenerate solutions.

In Fig. A.3, for each ROC curve, the point $(P_f(\tau_1^*, 0), P_d(\tau_1^*, 0))$ has been marked. This point is important for two reasons: 1) it determines how often the second detector in the cascade is triggered which, together with the activity level, determines the average energy consumption for the entire curve, and 2) it signifies the boundary of degenerate solutions. Visually, we see that for false-alarm rates greater than $P_f(\tau_1, 0)$, running the energy detector alone achieves a higher true-detection rate; hence, one is better off not running the cascade. This gives a fast visual method to determine which (β, α) constraints are degenerate.

A.3 Conclusions

We have studied and analyzed the problem of using the cascade architecture to control signal-model complexity for energy-aware detection. Our theoretical result states that under suitable conditions, utilizing all of the available energy and false alarm rate is optimal. This notion is not true in general and sufficient conditions for this property to hold are:

1. All detectors (i.e. cascade stages) operate on the *same* observation.

2. Detectors are the likelihood ratio tests that are derived from the assumed signal models.

Although our characterization of the optimal solution results in theoretically "easy" root-finding problems, practically speaking, finding optimal thresholds is still a hard problem because 1) even for simple problems, closed-form expressions for the cascade detection performance do not exist, and 2) in practice, signal-model parameters are not known exactly and are time-varying. Our solution to simultaneously handle both of these problems was to track the detector thresholds by averaging detector decisions. This adaptive technique can be useful for adjusting to a time-varying energy budget, which is relevant in energy-harvesting devices.

Finally, we demonstrated on the CheetahCub testbed how to synergistically exploit the hardware scalability. As a case study, we considered the design of a battery-powered wildlife monitoring system, resulting in 23x energy scalability. We discussed how to visualize the performance/energy trade-off, and demonstrated 3x to 20x energy savings with minimal loss in performance.

A.4 Proof of Theorem 2

We first review a result that follows from the Neyman-Pearson lemma [81]. Denote the triple $(\mathcal{Y}, \mathcal{F}_{\mathcal{Y}}, P)$ as a probability space. Let δ_α^* be the optimal Neyman-Pearson decision rule for a test of size of α, and $Y_{\delta_\alpha^*}$ as its detection region. Then,

1. The optimal δ_α^* is a likelihood ratio test (LRT).

2. The achieved false alarm rate is $P(Y_{\delta_\alpha^*} \mid H_0 \text{ true}) = \alpha$.

3. Finding the optimal decision rule δ_α^* is equivalent to specifying $Y_{\delta_\alpha^*}$. That is:

$$\begin{aligned} & \max_{\delta_\alpha \in \Delta} && P(Y_{\delta_\alpha} \mid H_1 \text{ true}) \\ & \text{s.t.} && P(Y_{\delta_\alpha} \mid H_0 \text{ true}) \leq \alpha \\ \iff & \max_{Y \in \mathcal{F}_{\mathcal{Y}}} && P(Y \mid H_1 \text{ true}) \\ & \text{s.t.} && P(Y \mid H_0 \text{ true}) \leq \alpha \end{aligned} \tag{A.13}$$

   where Δ is the space of all randomized decision rules.

Lemma 2 *The results from the NP lemma hold even when we restrict the search in (A.13) to a subset, $\bar{Y}$, of $\mathcal{Y}$ (and the corresponding σ-algebra generated by $\bar{Y}$, $\mathcal{F}_{\bar{Y}}$), given that $P(\bar{Y} \mid H_0 \text{ true}) \geq \alpha$. That is, given $(\bar{Y}, \mathcal{F}_{\bar{Y}}, P)$, let $\bar{\delta}_\alpha^*$ be the optimal solution when the observation space is restricted to $\bar{Y}$, and denote $Y_{\bar{\delta}_\alpha^*} = \{y \in \mathcal{Y} : \bar{\delta}_\alpha^*(y) = H_1\}$ as the detection region. Note, by construction, $Y_{\bar{\delta}_\alpha^*} \subset \bar{Y}$. Then,*

1. *The optimal $\bar{\delta}_\alpha^*$ is a likelihood ratio test (LRT).*

2. *The achieved false alarm rate is $P(Y_{\bar{\delta}_\alpha^*} \mid H_0 \text{ true}) = \alpha$.*

3. *Finding the optimal decision rule $\bar{\delta}_\alpha^*$ is equivalent to specifying $Y_{\bar{\delta}_\alpha^*}$.*

That is:

$$\begin{aligned} & \max_{\bar{\delta}_\alpha \in \bar{\Delta}} && P(Y_{\bar{\delta}_\alpha} \mid H_1 \textit{ true}) \\ & \textit{s.t.} && P(Y_{\bar{\delta}_\alpha} \mid H_0 \textit{ true}) \leq \alpha \\ \iff & \max_{Y \in \mathcal{F}_{\bar{Y}}} && P(Y \mid H_1 \textit{ true}) \\ & \textit{s.t.} && P(Y \mid H_0 \textit{ true}) \leq \alpha \end{aligned} \tag{A.14}$$

where $\bar{\Delta}$ is the space of randomized decision rules with domain restricted to $\bar{Y}$.

Proof Follow the proof of the NP Lemma, replacing $\mathcal{Y}$ with $\bar{Y}$. Because $\mathcal{F}_{\bar{Y}} \subset \mathcal{F}_{\mathcal{Y}}$ and because we have not changed the probability measure P, the analogous result follows. ∎

The next result states that for a LRT, lowering the threshold cannot decrease the true-detection and false-alarm rates.

Lemma 3 *Let δ^*_α be the optimal NP decision rule, $Y_{\delta^*_\alpha}$ the corresponding detection region, and τ^*_α a corresponding threshold. Then, we have the following:*

1. *For any $\tau_\alpha < \tau^*_\alpha$, $Y_{\delta^*_\alpha} \subset Y_{\delta_\alpha}$, where Y_{δ_α} is the detection region for τ_α.*
2. *$P(Y_{\delta^*_\alpha} \mid H_i \textit{ true}) \leq P(Y_{\delta_\alpha} \mid H_i \textit{ true})$ for $i = 0, 1$.*

Proof The first claim follows from the definition of $\subset$, which holds because of the definition of a LRT. (2) follows from the axioms of a probability measure. In particular, for a probability space $(\mathrm{Y}, \mathcal{F}_{\mathcal{Y}}, P)$, for any subsets A, B of $\mathcal{F}_{\mathcal{Y}}$:

$$A \subset B \implies P(A) \leq P(B) \tag{A.15}$$

∎

Proof of Theorem 2 The proof is done by considering two cases: 1) when $P_f(\tau_1^*, \tau_2^*) = \alpha$, and 2) when $P_f(\tau_1^*, \tau_2^*) < \alpha$:

1. Assume $P_f(\tau_1^*, \tau_2^*) = \alpha$ and $EC(\tau_1^*) < \beta$. This is the harder of the two cases because we must show that true-detection performance does not decrease, even when we *increase* the threshold of the second detector.

   (a) Set $\tau_1' < \tau_1^*$ such that $EC(\tau_1') = \beta$. Then, $P_f(\tau_1', \tau_2^*) \geq \alpha$.

   (b) We must show that we can find a $\tau_2' > \tau_2^*$ such that the false alarm constraint is satisfied, *and* the true-detection rate does not decrease.

      i. The trick to showing this is to *relax* the problem by searching for δ_2' over all decision rules on the subset $\bar{Y}_{\delta_1'}$.

   (c) Apply Lemma 1 twice: first, letting $\bar{Y} = Y_{\delta_1^*}$, denote δ_2^* as a solution. Then, letting $\bar{Y} = Y_{\delta_1'}$, denote δ_2' as a solution.

   (d) From Lemma 2, $\tau_1' < \tau_1^* \implies Y_{\delta_1^*} \subset Y_{\delta_1'}$. As a result, $\mathcal{F}_{Y_{\delta_1^*}} \subset \mathcal{F}_{Y_{\delta_1'}}$. Hence, because the maximum over a set cannot be smaller than the maximum over a subset, we have that:

   $$P(Y_{\delta_2'} \mid H_1 \text{ true}) \geq P(Y_{\delta_2^*} \mid H_1 \text{ true}) \tag{A.16}$$

   which follows from the equivalence in (A.14). Note that in general, $Y_{\delta_2^*} \not\subset Y_{\delta_2'}$.

   (e) Although we originally relaxed the problem and searched over all decision rules, from Lemma 1, we know that δ_2' will be a LRT (i.e.

τ_2' exists), and:

$$P_d(\tau_1^*, \tau_2^*) \leq P_d(\tau_1', \tau_2') \tag{A.17}$$

$$P_f(\tau_1^*, \tau_2^*) = P_f(\tau_1', \tau_2') = \alpha \tag{A.18}$$

$$EC(\tau_1^*) < EC(\tau_1') = \beta \tag{A.19}$$

where the first line follows from (A.16), the second line follows from the second property of Lemma 1, and the third line follows from our assumption about τ_1^* and our choice of τ_1'.

2. Assume $P_f(\tau_1^*, \tau_2^*) < \alpha$, and $EC(\tau_1^*) < \beta$. This case is easy because we only need to decrease τ_1^*, which by Lemma 2 increases the detection performance.

   (a) Let $\tau_1' = \tau_1^* - \epsilon$, where $\epsilon > 0$ is chosen such that (τ_1', τ_2^*) is feasible and: $P_f(\tau_1', \tau_2^*) = \alpha$ or $EC(\tau_1') = \beta$. Such an ϵ exists due to Assumption (A7) (i.e. no point-mass assumption).

   (b) If $P_f(\tau_1', \tau_2^*) = \alpha$ and $EC(\tau_1') < \beta$, then this falls under case 1, and we are done.

   (c) If $EC(\tau_1') = \beta$, then $P_f(\tau_1^*, \tau_2^*) \leq P_f(\tau_1', \tau_2^*) \leq \alpha$, and $P_d(\tau_1^*, \tau_2^*) \leq P_d(\tau_1', \tau_2^*)$, which follows from Lemma 2.

∎

REFERENCES

[1] M. Buettner, B. Greenstein, A. Sample, J. R. Smith, and D. Wetherall, "Revisiting smart dust with RFID sensor networks," in Proc. 7th ACM Workshop on Hotnets, Calgary, Alberta, Canada, Oct. 2008.

[2] D. J. Yeager, P. S. Powledge, R. Prasad, D. Wetherall, and J. R. Smith, "Wirelessly-charged UHF tags for sensor data collection," in IEEE International Conference on RFID, 2008, pp. 320–327.

[3] H. Noguchi, T. Takagi, M. Yoshimoto, and H. Kawaguchi, "An ultra-low-power VAD hardware implementation for intelligent ubiquitous sensor networks," in IEEE Workshop on Signal Processing Systems, Oct. 2009, pp. 214 –219.

[4] D. H. Goldberg, A. G. Andreou, P. Julián, P. O. Pouliquen, L. Riddle, and R. Rosasco, "VLSI implementation of an energy-aware wake-up detector for an acoustic surveillance sensor network," ACM Trans. Sen. Netw., vol. 2, no. 4, pp. 594–611, 2006.

[5] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a wireless sensor network platform for detecting rare, random, and ephemeral events," in Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, 2005, p. 70.

[6] A. Benbasat and J. Paradiso, "A framework for the automated generation of power-efficient classifiers for embedded sensor nodes," in SenSys '07: Proceedings of the 5th Int. Conference on Embedded Networked Sensor Systems, 2007, pp. 219–232.

[7] N. Rajput and A. Nanavati, Speech in Mobile and Pervasive Environments. Wiley, 2012.

[8] S. Fine, Y. Singer, and N. Tishby, "The hierarchical hidden markov model: Analysis and applications," Machine learning, vol. 32, no. 1, pp. 41–62, 1998.

[9] R. Neugebauer and D. Mcauley, "Energy is just another resource: energy accounting and energy pricing in the Nemesis OS," in Hot Topics in

Operating Systems, 2001. Proceedings of the Eighth Workshop on, 2001, pp. 67–72.

[10] A. Hero, Foundations and applications of sensor management. Springer, 2008.

[11] D. Bertsekas, Dynamic programming and optimal control. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[12] P. R. Kumar and P. Varaiya, Stochastic Systems: Estimation, Identification and Adaptive Control. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1986.

[13] R. Smallwood and E. Sondik, "The optimal control of partially observable markov processes over a finite horizon," Operations Research, vol. 21, no. 5, pp. 1071–1088, 1973.

[14] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," Artificial intelligence, vol. 101, no. 1, pp. 99–134, 1998.

[15] A. Cassandra, M. L. Littman, and N. L. Zhang, "Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes," in Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1997, pp. 54–61.

[16] A. R. Cassandra, "POMDP solver software." [Online]. Available: http://www.pomdp.org/pomdp/code/index.shtml

[17] E. Sondik, "The optimal control of partially observable markov processes over the infinite horizon: Discounted costs," Operations Research, vol. 26, no. 2, pp. 282–304, 1978.

[18] M. Hauskrecht, "Value-function approximations for partially observable markov decision processes," Journal of A.I. Research, 2000.

[19] J. D. Williams and S. Young, "Scaling POMDPs for spoken dialog management," Audio, Speech, and Language Processing, IEEE Transactions on, vol. 15, no. 7, pp. 2116–2129, 2007.

[20] E. Hansen, "Solving pomdps by searching in policy space," in Proceedings of the fourteenth conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1998, pp. 211–219.

[21] N. Roy, G. Gordon, and S. Thrun, "Finding approximate POMDP solutions through belief compression," J. Artif. Intell. Res. (JAIR), vol. 23, pp. 1–40, 2005.

[22] P. Poupart, C. Boutilier et al., "Value-directed compression of POMDPs," Advances in Neural Information Processing Systems, vol. 15, pp. 1547–1554, 2002.

[23] M. Spaan and N. Spaan, "A point-based POMDP algorithm for robot planning," in IEEE International Conference on Robotics and Automation, 2004.

[24] J. Pineau, G. Gordon, S. Thrun et al., "Point-based value iteration: An anytime algorithm for pomdps," in International Joint Conference on Artificial Intelligence, vol. 18, 2003, pp. 1025–1032.

[25] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," Autonomous Agents and Multi-Agent Systems, 2012.

[26] M. T. Spaan and N. A. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs." J. Artif. Intell. Res.(JAIR), vol. 24, pp. 195–220, 2005.

[27] T. Smith and R. G. Simmons, "Heuristic search value iteration for POMDPs," in Proc. Int. Conf. on Uncertainty in A.I., 2004.

[28] P. Poupart, K.-E. Kim, and D. Kim, "Closing the gap: Improved bounds on optimal POMDP solutions." in ICAPS, 2011.

[29] J. Hoey and P. Poupart, "Solving POMDPs with continuous or large discrete observation spaces," in International Joint Conference on Artificial Intelligence, vol. 19. LAWRENCE ERLBAUM ASSOCIATES LTD, 2005, p. 1332.

[30] J. M. Porta, M. T. Spaan, and N. Vlassis, "Robot planning in partially observable continuous domains," in In Robotics: Science and Systems I, 2005.

[31] J. Porta, "C-POMDP: POMDPs in continuous spaces." [Online]. Available: http://www.iri.upc.edu/people/porta/

[32] A. Hero and D. Cochran, "Sensor management: Past, present, and future," Sensors Journal, IEEE, vol. 11, no. 12, pp. 3064–3075, 2011.

[33] E. K. Chong, C. M. Kreucher, and A. O. Hero, Iii, "Partially observable Markov decision process approximations for adaptive sensing," Discrete Event Dynamic Systems, vol. 19, no. 3, pp. 377–422, Sep. 2009.

[34] J. Fuemmeler, "Energy-efficient tracking in sensor networks," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2008.

[35] K. A. Harris and V. V. Veeravalli, "Implementing energy-efficient tracking in a sensor network," in IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013, pp. 4608–4612.

[36] D. S. Zois, M. Levorato, and U. Mitra, "Energy-efficient, heterogeneous sensor selection for physical activity detection in wireless body area networks," Signal Processing, IEEE Transactions on, vol. 61, no. 7, pp. 1581–1594, April 2013.

[37] G. Atia, V. Veeravalli, and J. Fuemmeler, "Sensor scheduling for energy-efficient target tracking in sensor networks," Signal Processing, IEEE Transactions on, vol. 59, no. 10, pp. 4923–4937, Oct 2011.

[38] V. Krishnamurthy, "Algorithms for optimal scheduling and management of hidden Markov model sensors," Signal Processing, IEEE Transactions on, vol. 50, no. 6, pp. 1382–1397, 2002.

[39] V. Krishnamurthy and D. Djonin, "Structured threshold policies for dynamic sensor schedulinga partially observed Markov decision process approach," Signal Processing, IEEE Transactions on, vol. 55, no. 10, pp. 4938–4957, 2007.

[40] D. Castañon, "Approximate dynamic programming for sensor management," in Decision and Control, 1997., Proceedings of the 36th IEEE Conference on, vol. 2. IEEE, 1997, pp. 1202–1207.

[41] D. Hitchings and D. Castañon, "Receding horizon stochastic control algorithms for sensor management," in American Control Conference (ACC), 2010. IEEE, 2010, pp. 6809–6815.

[42] D. Weiss, B. Sapp, and B. Taskar, "Dynamic structured model selection," in Proceedings of the International Conference on Computer Vision (ICCV), 2013.

[43] J. Williams, "Information theoretic sensor management," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[44] H. Yu and D. P. Bertsekas, "Discretized approximations for POMDP with average cost," in Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, ser. UAI '04. Arlington, Virginia, United States: AUAI Press, 2004, pp. 619–627.

[45] D. M. Jun, D. L. Jones, T. P. Coleman, W. J. Leonard, and R. Ratnam, "Practical sensor management for an energy-limited detection system," in IEEE International Conference on Acoustics, Speech and Signal Processing, 2012.

[46] V. Krishnamurthy, "How to schedule measurements of a noisy Markov chain in decision making?" Information Theory, IEEE Transactions on, vol. 59, no. 7, pp. 4440–4461, July 2013.

[47] S. Gorantla, "The interplay between information and control theory within interactive decision-making problems," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2012.

[48] J. Evans and V. Krishnamurthy, "Optimal sensor scheduling for hidden markov model state estimation," International Journal of Control, vol. 74, no. 18, pp. 1737–1742, 2001.

[49] D. Jun, D. Cohen, and D. Jones, "A direct algorithm for joint optimal sensor scheduling and MAP state estimation for hidden Markov models," in IEEE International Conference on Acoustics, Speech and Signal Processing, 2013.

[50] M. Hauskrecht, "Incremental methods for computing bounds in partially observable markov decision processes," in Nat. Conf. A.I., 1997.

[51] J. R. Norris, Markov Chains. Cambridge University Press, 1998.

[52] W. J. Leonard, J. Neal, and R. Ratnam, "Variation of type B song in the endangered golden-cheeked warbler (dendroica chrysoparia)," The Wilson Journal of Ornithology, vol. 122, no. 4, 2010.

[53] J. S. Bolsinger, "Use of two song categories by golden-cheeked warblers," The Condor, vol. 102, no. 3, pp. 539–552, 2000.

[54] B. Levy, Principles of Signal Detection and Parameter Estimation. New York, NY: Springer, 2008.

[55] P. Dutta, M. Feldmeier, J. Paradiso, and D. Culler, "Energy metering for free: Augmenting switching regulators for real-time monitoring," in Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on. IEEE, 2008, pp. 283–294.

[56] Energy Micro, "User manual starter kit efm32tg-stk3300," 2011.

[57] L. Le, "Energy-efficient detection system in time-varying signal and noise power," M.S. thesis, University of Illinois at Urbana-Champaign, 2013.

[58] J. Yiu and A. Frame, "32-bit microcontroller code size analysis," 2013. [Online]. Available: http://www.arm.com/files/pdf/ARM_Microcontroller_Code_Size_(full).pdf

[59] Analog Devices, "ADMP801 datasheet: Hearing aid omnidirection MEMS microphone," Jun 2013.

[60] M. Malinowski, M. Moskwa, M. Feldmeier, M. Laibowitz, and J. A. Paradiso, "Cargonet: a low-cost micropower sensor node exploiting quasi-passive wakeup for adaptive asychronous monitoring of exceptional events," in Proceedings of the 5th international conference on Embedded networked sensor systems. ACM, 2007, pp. 145–159.

[61] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, 1989.

[62] S. Z. Yu, "Hidden semi-Markov models," Artificial Intelligence, vol. 174, no. 2, pp. 215–243, 2010.

[63] K. P. Murphy, "Dynamic Bayesian networks: representation, inference and learning," Ph.D. dissertation, University of California, 2002.

[64] T. Smith, "ZMDP software for POMDP and MDP planning." [Online]. Available: https://github.com/trey0/zmdp

[65] D. Jun and D. L. Jones, "Cascading signal-model complexity for energy-aware detection," IEEE Journal Emerging Topics CAS, 2013.

[66] S. Ross, B. Chaib-draa, and J. Pineau, "Bayes-adaptive POMDPs," in Advances in neural information processing systems, 2007, pp. 1225–1232.

[67] P. Greenhalgh, "Big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7," Sept 2011.

[68] A. Sinha, A. Wang, and A. Chandrakasan, "Energy scalable system design," IEEE Trans. on VLSI Systems, vol. 10, no. 2, pp. 135 –145, April 2002.

[69] J. Chen, R. Tan, G. Xing, X. Wang, and X. Fu, "Fidelity-aware utilization control for cyber-physical surveillance systems," in Proceedings of the 2010 31st IEEE Real-Time Systems Symposium, ser. RTSS '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 117–126.

[70] V. C. Raykar, B. Krishnapuram, and S. Yu, "Designing efficient cascaded classifiers: Tradeoff between accuracy and cost," in Proceedings of the 16th ACM SIGKDD. ACM, 2010, pp. 853–860.

[71] H. Luo, "Optimization design of cascaded classifiers," in IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, vol. 1, 2005, pp. 480 – 485.

[72] P. Varshney and C. Burrus, Distributed Detection and Data Fusion, ser. Signal processing and digital filtering. Springer, 1997.

[73] E. Ertin, "Polarimetric processing and sequential detection for automatic target recognition systems," Ph.D. dissertation, The Ohio State University, 1999.

[74] S. Nawab, A. Oppenheim, A. Chandrakasan, J. Winograd, and J. Ludwig, "Approximate signal processing," J. VLSI Signal Process. Syst., vol. 15, no. 1/2, pp. 177–200, 1997.

[75] J. Winograd, S. Nawab, and A. Oppenheim, "FFT-based incremental refinement of suboptimal detection," in IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 5, May 1996, pp. 2479 –2482.

[76] Y. Andreopoulos and I. Patras, "Incremental refinement of image salient-point detection," IEEE Trans. on Image Processing, vol. 17, no. 9, pp. 1685 –1699, Sept. 2008.

[77] M. Goel and N. Shanbhag, "Dynamic algorithm transformations (DAT)-a systematic approach to low-power reconfigurable signal processing," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 7, no. 4, pp. 463 –476, Dec. 1999.

[78] D. M. Jun and D. L. Jones, "An energy-aware framework for cascaded detection algorithms," in IEEE Workshop on Signal Processing Systems, Oct. 2010, pp. 1–6.

[79] H. Robbins and S. Monro, "A stochastic approximation method," Annals of Mathematical Statistics, vol. 22, pp. 400–407, 1951.

[80] H. Kushner and G. Yin, Stochastic Approximation and Recursive Algorithms and Applications. Springer, 2003.

[81] T. S. Ferguson, Mathematical Statistics: A Decision Theoretic Approach. Boston: Academic Press, 1967.